

CENTRALNA KOMISJA EGZAMINACYJNA
OKRĘGOWE KOMISJE EGZAMINACYJNE

INFORMATOR
O EGZAMINIE MATURALNYM
Z INFORMATYKI
OD ROKU SZKOLNEGO 2014/2015

INFORMATOR O EGZAMINIE MATURALNYM Z INFORMATYKI OD ROKU SZKOLNEGO 2014/2015

opracowany przez Centralną Komisję Egzaminacyjną
we współpracy z okręgowymi komisjami egzaminacyjnymi
w Gdańsku, Jaworznie, Krakowie, Łodzi,
Łomży, Poznaniu, Warszawie i we Wrocławiu



Centralna Komisja Egzaminacyjna
Warszawa 2013

Centralna Komisja Egzaminacyjna

ul. Józefa Lewartowskiego 6, 00-190 Warszawa

tel. 22 536 65 00

ckesekr@cke.edu.pl

Okręgowa Komisja Egzaminacyjna w Gdańsku

ul. Na Stoku 49, 80-874 Gdańsk

tel. 58 320 55 90

komisja@oke.gda.pl

Okręgowa Komisja Egzaminacyjna w Jaworznie

ul. Adama Mickiewicza 4, 43-600 Jaworzno

tel. 32 616 33 99

oke@oke.jaworzno.pl

Okręgowa Komisja Egzaminacyjna w Krakowie

os. Szkolne 37, 31-978 Kraków

tel. 12 683 21 01

oke@oke.krakow.pl

Okręgowa Komisja Egzaminacyjna w Łomży

ul. Nowa 2, 18-400 Łomża

tel. 86 216 44 95

sekretariat@oke.lomza.pl

Okręgowa Komisja Egzaminacyjna w Łodzi

ul. Ksawerego Praussa 4, 94-203 Łódź

tel. 42 634 91 33

komisja@komisja.pl

Okręgowa Komisja Egzaminacyjna w Poznaniu

ul. Gronowa 22, 61-655 Poznań

tel. 61 854 01 60

sekretariat@oke.poznan.pl

Okręgowa Komisja Egzaminacyjna w Warszawie

ul. Grzybowska 77, 00-844 Warszawa

tel. 22 457 03 35

info@oke.waw.pl

Okręgowa Komisja Egzaminacyjna we Wrocławiu

ul. Tadeusza Zielińskiego 57, 53-533 Wrocław

tel. 71 785 18 94

sekretariat@oke.wroc.pl

Spis treści

Wstęp	7
1. Opis egzaminu maturalnego z informatyki na poziomie rozszerzonym	9
1.1. Zakres wiadomości i umiejętności sprawdzanych na egzaminie	9
1.2. Ogólne informacje o egzaminie maturalnym z informatyki od roku szkolnego 2014/2015	9
1.3. Arkusz egzaminacyjny z informatyki na poziomie rozszerzonym	10
1.4. Ocenianie odpowiedzi zdających	10
2. Przykładowe zadania z informatyki na poziomie rozszerzonym wraz z rozwiązaniami	13
Opinia Konferencji Rektorów Akademickich Szkół Polskich o informatorach maturalnych od 2015 roku	81

Wstęp

Informator o egzaminie maturalnym z informatyki od roku szkolnego 2014/2015 jest podzielony na dwie części.

CZĘŚĆ PIERWSZA (1.1.–1.4.) zawiera ogólne informacje dotyczące egzaminu maturalnego z informatyki, w tym zakres sprawdzanych wiadomości i umiejętności, krótką charakterystykę arkusza egzaminacyjnego oraz sposobu oceniania odpowiedzi w zadaniach zamkniętych i otwartych.

CZĘŚĆ DRUGA zawiera przykładowe zadania z informatyki, jakie mogą pojawić się w arkuszach egzaminacyjnych w obu częściach egzaminu. Do każdego zadania:

- przypisano najważniejsze wymagania ogólne i szczegółowe z podstawy programowej kształcenia ogólnego, do których to zadanie się odnosi,
- podano przykładowe rozwiązania zadań otwartych oraz odpowiedzi do zadań zamkniętych.

Zadania w Informatorze:

- nie wyczerpują wszystkich typów zadań, które mogą wystąpić w arkuszach egzaminacyjnych,
- nie ilustrują wszystkich wymagań z zakresu informatyki w podstawie programowej,
- nie zawierają wszystkich możliwych rodzajów materiałów źródłowych, które mogą stanowić obudowę zadań.

Informator nie może być zatem jedyną ani nawet główną wskazówką do planowania procesu kształcenia w zakresie informatyki w szkole ponadgimnazjalnej. Tylko realizacja wszystkich wymagań z podstawy programowej może zapewnić wszechstronne wykształcenie uczniów szkół ponadgimnazjalnych.

Przed przystąpieniem do dalszej lektury *Informatora* warto zapoznać się z ogólnymi zasadami obowiązującymi na egzaminie maturalnym od roku szkolnego 2014/2015. Są one określone w rozporządzeniu Ministra Edukacji Narodowej z dnia 30 kwietnia 2007 r. w sprawie warunków i sposobu oceniania, klasyfikowania i promowania uczniów i słuchaczy oraz sposobu przeprowadzania sprawdzianów i egzaminów w szkołach publicznych (Dz.U. nr 83, poz. 562, z późn. zm.), w tym w szczególności w rozporządzeniu z 25 kwietnia 2013 r. zmieniającym powyższe rozporządzenie (Dz.U. z 2013 r., poz. 520), oraz – w skróconej formie – w części ogólnej *Informatora o egzaminie maturalnym od roku szkolnego 2014/2015*, dostępnej na stronie internetowej Centralnej Komisji Egzaminacyjnej (www.cke.edu.pl) oraz na stronach internetowych okręgowych komisji egzaminacyjnych.

1.

Opis egzaminu maturalnego z informatyki na poziomie rozszerzonym

1.1. Zakres wiadomości i umiejętności sprawdzanych na egzaminie

Egzamin maturalny z informatyki sprawdza, w jakim stopniu absolwent spełnia wymagania z zakresu tego przedmiotu określone w podstawie programowej kształcenia ogólnego dla IV etapu edukacyjnego w zakresie rozszerzonym i podstawowym. Poszczególne zadania zestawu egzaminacyjnego mogą też odnosić się do wymagań przypisanych do etapów wcześniejszych, tj. II (klasy 4–6 szkoły podstawowej) oraz III (gimnazjum).

Podstawa programowa dzieli wymagania na szczegółowe i ogólne. Wymagania szczegółowe nie są hasłami odnoszącymi się do całościowych obszarów wiedzy, lecz odwołują się do ściśle określonych wiadomości i konkretnych umiejętności. Wymagania ogólne, jako syntetyczne ujęcie nadrzędnych celów kształcenia, stanowiące odpowiedź na pytanie, po co uczymy informatyki, informują, jak rozumieć podporządkowane im wymagania szczegółowe. Sposób spełniania wymagań szczegółowych jest wartościowy tylko wtedy, gdy przybliży osiągnięcie celów zawartych w wymaganiach ogólnych.

Zadania w arkuszu maturalnym z informatyki na poziomie rozszerzonym mają na celu sprawdzenie w szczególności:

- znajomości i umiejętności posługiwania się komputerem i jego oprogramowaniem oraz korzystania z sieci komputerowych,
- umiejętności wyszukiwania, gromadzenia, selekcionowania, przetwarzania i wykorzystywania informacji, zasad współtworzenia zasobów w sieci, korzystania z różnych źródeł oraz znajomości sposobów zdobywania informacji,
- znajomości i umiejętności komunikowania się za pomocą komputera i technologii informacyjno-komunikacyjnych,
- umiejętności opracowywania informacji za pomocą komputera, w tym rysunków, tekstów, danych liczbowych, animacji, prezentacji multimedialnych i filmów,
- umiejętności rozwiązywania problemów i podejmowania decyzji z wykorzystaniem komputera, stosowania podejścia algorytmicznego,
- znajomości i umiejętności stosowania podstawowych algorytmów.

W informatorze dla każdego zadania podano najważniejsze wymagania ogólne i szczegółowe, do których to zadanie się odnosi, oraz zamieszczono przykładowe rozwiązanie/rozwiązania wraz z komentarzem.

1.2. Ogólne informacje o egzaminie maturalnym z informatyki od roku szkolnego 2014/2015

Od roku szkolnego 2014/2015¹ egzamin maturalny z informatyki może być zdawany wyłącznie jako przedmiot dodatkowy na poziomie rozszerzonym.

Egzamin ma formę pisemną, trwa 210 minut i składa się z dwóch części:

- a) część pierwsza trwa 60 minut i polega na rozwiązaniu zestawu zadań bez korzystania z komputera,
- b) część druga trwa 150 minut i polega na rozwiązaniu zadań przy użyciu komputera.

¹ W przypadku absolwentów techników – od roku szkolnego 2015/2016.

Do egzaminu z informatyki może przystąpić każdy absolwent, niezależnie od typu szkoły, do której uczęszczał, oraz od przedmiotów, których uczył się w zakresie rozszerzonym. W czasie trwania pierwszej części egzaminu zdający może korzystać z kalkulatora.

Wyniki części pisemnej egzaminu maturalnego są wyrażane w procentach i na skali centylowej (por. punkt G. „Ocenianie i wyniki egzaminu” w CZEŚCI OGÓLNEJ *Informatora o egzaminie maturalnym od roku szkolnego 2014/2015*). Wyniki uzyskane w części pisemnej egzaminu maturalnego z informatyki – podobnie jak z innych przedmiotów dodatkowych – nie mają wpływu na zdanie egzaminu maturalnego.²

1.3. Arkusz egzaminacyjny z informatyki na poziomie rozszerzonym

Arkusze egzaminacyjne do każdej części egzaminu z informatyki będą zawierały około 4 zadań. W drugiej części egzaminu zdający będzie pracował na autonomicznym stanowisku komputerowym i będzie mógł korzystać wyłącznie z programów i danych zapisanych na dysku twardym oraz na innych nośnikach stanowiących wyposażenie stanowiska lub otrzymanych z arkuszem egzaminacyjnym. Przy numerze każdego zadania podana będzie maksymalna liczba punktów, którą można uzyskać za poprawne jego rozwiązanie.

Zadania w arkuszu egzaminacyjnym:

- będą dobrane w taki sposób, aby reprezentowały różnorodne wymagania ogólne i szczegółowe z podstawy programowej,
- będą sprawdzały przede wszystkim umiejętności złożone, w tym np. umiejętność informatycznego rozwiązywania problemu, charakteryzująca się przestrzeganiem kolejności etapów rozwiązania: od projektowania do otrzymania rozwiązania,
- będą sprawdzały umiejętności rozwiązywania postawionego problemu na podstawie informacji przedstawionych w różnej formie oraz umiejętności ich przetwarzania i analizowania,
- będą zróżnicowane pod względem poziomu trudności oraz sposobu udzielania odpowiedzi,
- będą miały formę zamkniętą lub otwartą. W zadaniach zamkniętych, np. wyboru wielokrotnego, prawda / fałsz, na dobieranie, zdający wybiera jedną z podanych opcji odpowiedzi, natomiast w zadaniach otwartych – tworzy odpowiedź samodzielnie. W arkuszu będą przeważały zadania otwarte.
- będą występowały pojedynczo lub w wiązkach tematycznych,
- będą odnosić się do różnorodnych materiałów źródłowych zamieszczonych w arkuszu, np. zwartych fragmentów artykułów popularnonaukowych, algorytmów, schematów, tabel z danymi.

1.4. Ocenianie odpowiedzi zdających

Rozwiązania poszczególnych zadań oceniane są na podstawie szczegółowych kryteriów oceniania, jednolitych w całym kraju. Egzaminatorzy w szczególności zwracają uwagę na poprawność merytoryczną rozwiązań, kompletność i dokładność prezentacji rozwiązań zadań, tworzonych dokumentów, zachowanie odpowiednich zasad w zapisie programów i algorytmów. Ocenianiu podlegają tylko te fragmenty pracy zdającego, które dotyczą polecenia. Komentarze i rozwiązania, nawet poprawne, nie mające związku z poleceniem nie podlegają ocenianiu.

² Z wyjątkiem sytuacji, kiedy egzamin z informatyki był jedynym egzaminem z przedmiotu dodatkowego, którego zdawanie zadeklarował zdający, po czym nie przystąpił do tego egzaminu lub egzamin ten został mu unieważniony.

Za rozwiązanie zadania, którego celem jest ułożenie i napisanie algorytmu, zdający otrzymuje pełną liczbę punktów tylko za poprawny i działający algorytm. W przypadku usterek, np. niepoprawnej konstrukcji pętli, pominięcia niektórych elementów w zliczaniu, błędów w przeszukiwaniu tablicy, indeksowaniu – przyznawana jest tylko część punktów (za poprawnie zapisane czynności wynikające z treści zadania).

W zadaniach praktycznych, w drugiej części egzaminu, oceniane są rzeczywiste efekty i osiągnięte rezultaty pracy zdającego, np. wyniki obliczeń w arkuszu kalkulacyjnym, wyniki symulacji, odpowiedzi uzyskane za pomocą kwerend, wyniki uzyskane za pomocą programu napisanego przez zdającego. Dane do zadań programistycznych mogą być tak dobrane, że uzyskanie wyników dla danych dużych rozmiarów będzie wymagać zastosowania w napisanym programie algorytmu o jak najmniejszej złożoności.

Zdający w drugiej części egzaminu, jako rozwiązanie zadania, powinien przekazać do oceny pliki zawierające komputerową realizację rozwiązania/obliczeń oraz pliki (najczęściej tekstowe) zawierające odpowiedzi do zadania/zadań.

Za całkowicie poprawne rozwiązania zadań, uwzględniające inny tok rozumowania niż podany w kryteriach oceniania, przyznawana jest pełna liczba punktów.

Gdy do jednego polecenia zdający podaje kilka rozwiązań (prawidłowe i błędne), to egzaminator nie wybiera prawidłowego rozwiązania i nie przyznaje punktów.

2.

Przykładowe zadania z informatyki na poziomie rozszerzonym
wraz z rozwiązaniami

Zadania 1–12 zawierają po cztery odpowiedzi, z których każda jest albo prawdziwa, albo fałszywa. Zdecyduj, które z podanych odpowiedzi są prawdziwe (**P**), a które fałszywe (**F**). Zaznacz znakiem X odpowiednią rubrykę w tabeli.

Zadanie 1. (0–1)

Dana jest tabela:

Sprawdzian

uczen	klasowka	egzamin
Abacki	45	0
Babacki	50	80
Cabacki	100	90
Dabacki	80	70

Dla powyższej tabeli utworzono następujące zapytanie w SQL:

```
SELECT uczen
FROM Sprawdzian
WHERE (klasowka > egzamin AND egzamin > 75) OR klasowka < 50
```

Wynikiem tego zapytania jest

	P	F
Abacki, Babacki.		
Babacki, Cabacki.		
Abacki, Cabacki.		
Abacki, Dabacki.		

Wymagania ogólne	<i>II. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera.</i>
Wymagania szczegółowe	<i>2. Wyszukiwanie, gromadzenie, selekcjonowanie, przetwarzanie i wykorzystywanie informacji, korzystanie z różnych źródeł i sposobów zdobywania informacji. Zdający: 2) stosuje metody wyszukiwania i przetwarzania informacji w relacyjnej bazie danych.</i>
Rozwiązanie	FFPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 2. (0–1)Liczba $CB_{(16)}$ jest równa liczbie

	P	F
$1010101111_{(2)}$.		
$313_{(8)}$.		
$112011120_{(3)}$.		
$203_{(10)}$.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Zdający: 11) opisuje podstawowe algorytmy.</i>
Rozwiązanie	FPFP
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 3. (0–1)

W grafice rastrowej

	P	F
każdy piksel ma jednoznacznie określony kolor.		
obraz pamiętany jest w postaci obiektów geometrycznych.		
zaletą jest skalowalność obrazu.		
mogą być zapisywane zdjęcia z aparatu cyfrowego.		

Wymagania ogólne	<i>I. Bezpieczne posługiwanie się komputerem i jego oprogramowaniem, wykorzystanie sieci komputerowej; komunikowanie się za pomocą komputera i technologii informacyjno-komunikacyjnych.</i>
Wymagania szczegółowe	<i>1. Posługiwanie się komputerem i jego oprogramowaniem, korzystanie z sieci komputerowej. Zdający: 1) przedstawia sposoby reprezentowania różnych form informacji w komputerze: liczb, znaków, obrazów, animacji, dźwięków.</i>
Rozwiązanie	PFFP
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 4. (0–1)

Do szyfrowania informacji służy

	P	F
algorytm RSA.		
metoda bisekcji.		
PGP.		
algorytm Huffmana.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Zdający: 11) opisuje podstawowe algorytmy.</i>
Rozwiązanie	PFPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 5. (0–1)

Dynamicznym przydzielaniem numerów IP w sieci zajmuje się serwer

	P	F
DNS.		
DHCP.		
SMTP.		
FTP.		

Wymagania ogólne	<i>I. Bezpieczne posługiwanie się komputerem i jego oprogramowaniem, wykorzystanie sieci komputerowej; komunikowanie się za pomocą komputera i technologii informacyjno-komunikacyjnych.</i>
Wymagania szczegółowe	<i>1. Posługiwanie się komputerem i jego oprogramowaniem, korzystanie z sieci komputerowej. Zdający: 3) prawidłowo posługuje się terminologią sieciową.</i>
Rozwiązanie	FPPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 6. (0–1)

Dane: n – liczba naturalna większa od zera

Funkcja $K(n)$

1. dla $n < 4$ wynikiem jest 1
2. dla $n \geq 4$ wynikiem jest $K(n-1) - K(n-3)$

Dla funkcji K zachodzi

	P	F
dla każdego $n > 4$ zachodzi $K(n) < 0$.		
$K(2) > K(5)$.		
$K(10) = 3$.		
funkcja jest niemalejąca.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego. Zdający: 16) opisuje właściwości algorytmów na podstawie ich analizy.</i>
Rozwiązanie	FPPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 7. (0–1)

Rozważ poniższy algorytm, gdzie n jest liczbą całkowitą nieujemną.

- (1) $wynik \leftarrow 0$;
- (2) **dopóki** $n \neq 0$ **wykonuj**
- (3) $wynik \leftarrow wynik + (n \bmod 10)$
- (4) $n \leftarrow n \div 10$

gdzie: mod to operator reszty z dzielenia,
div to operator dzielenia całkowitego.

Dla podanego algorytmu zachodzi

	P	F
dla $n = 36789$ $wynik = 30$.		
dla $n = 11111111$ $wynik = 8$.		
$wynik$ jest równy sumie cyfr w zapisie dziesiętnym liczby n .		
dla $n = 1234$ zmienna $wynik$ w kolejnych iteracjach przyjmuje wartości 1, 3, 6, 10.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>16) opisuje właściwości algorytmów na podstawie ich analizy.</i>
Rozwiązanie	FPPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 8. (0–1)

Rozważ poniższy algorytm, gdzie n jest liczbą całkowitą nieujemną, $a[0..n]$ jest tablicą liczb całkowitych, z – liczbą rzeczywistą.

- (1) $i \leftarrow n; y \leftarrow a[n];$
- (2) **dopóki** $i \neq 0$ **wykonuj**
- (3) $i \leftarrow i - 1$
- (4) $y \leftarrow y * z + a[i]$

Algorytm ten przedstawia realizację

	P	F
obliczania wartości wielomianu dla danej wartości z .		
obliczenia NWW dla n liczb naturalnych.		
obliczenia NWD dla n liczb naturalnych.		
schematu Hornera.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>11) opisuje podstawowe algorytmy.</i>
Rozwiązanie	PFFP
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 9. (0–1)

Program komputerowy na licencji *freeware* można

	P	F
rozpowszechniać, jednak z zachowaniem informacji o autorze.		
wykorzystać do tworzenia nowych programów przez wprowadzanie w nim zmian.		
stosować do obliczeń.		
sprzedawać.		

Wymagania ogólne	<i>V. Ocena zagrożeń i ograniczeń, docenianie społecznych aspektów rozwoju i zastosowań informatyki.</i>
Wymagania szczegółowe	<i>7. Wykorzystywanie komputera i technologii informacyjno-komunikacyjnych do rozwijania zainteresowań, opisywanie zastosowań informatyki, ocena zagrożeń i ograniczeń, aspekty społeczne rozwoju i zastosowań informatyki.</i> Zdający: 3) stosuje normy etyczne i prawne związane z rozpowszechnianiem programów komputerowych, bezpieczeństwem i ochroną danych oraz informacji w komputerze i w sieciach komputerowych.
Rozwiązanie	PFPF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 10. (0–1)

Zgodnie z przepisami prawa autorskiego dozwolone jest

	P	F
publikowanie pod własnym nazwiskiem, na swojej stronie WWW, skopiowanych zasobów internetowych (zdjęć i artykułów).		
zamieszczanie na własnej stronie linków do innych stron WWW.		
zamieszczanie na własnej stronie cudzych programów na licencji <i>freeware</i> .		
zamieszczenie na stronie internetowej treści utworów, do których wygasły majątkowe prawa autorskie.		

Wymagania ogólne	<i>V. Ocena zagrożeń i ograniczeń, docenianie społecznych aspektów rozwoju i zastosowań informatyki.</i>
Wymagania szczegółowe	<i>7. Wykorzystywanie komputera i technologii informacyjno-komunikacyjnych do rozwijania zainteresowań, opisywanie zastosowań informatyki, ocena zagrożeń i ograniczeń, aspekty społeczne rozwoju i zastosowań informatyki.</i> Zdający: 3) stosuje normy etyczne i prawne związane z rozpowszechnianiem programów komputerowych, bezpieczeństwem i ochroną danych oraz informacji w komputerze i w sieciach komputerowych.
Rozwiązanie	FPPP
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 11. (0–1)

Do jednoznacznej identyfikacji osoby podpisującej cyfrowy dokument służy

	P	F
podpis elektroniczny.		
wpisanie imienia i nazwiska.		
zaszyfrowanie dokumentu.		
wstawienie zeskanowanego podpisu autora.		

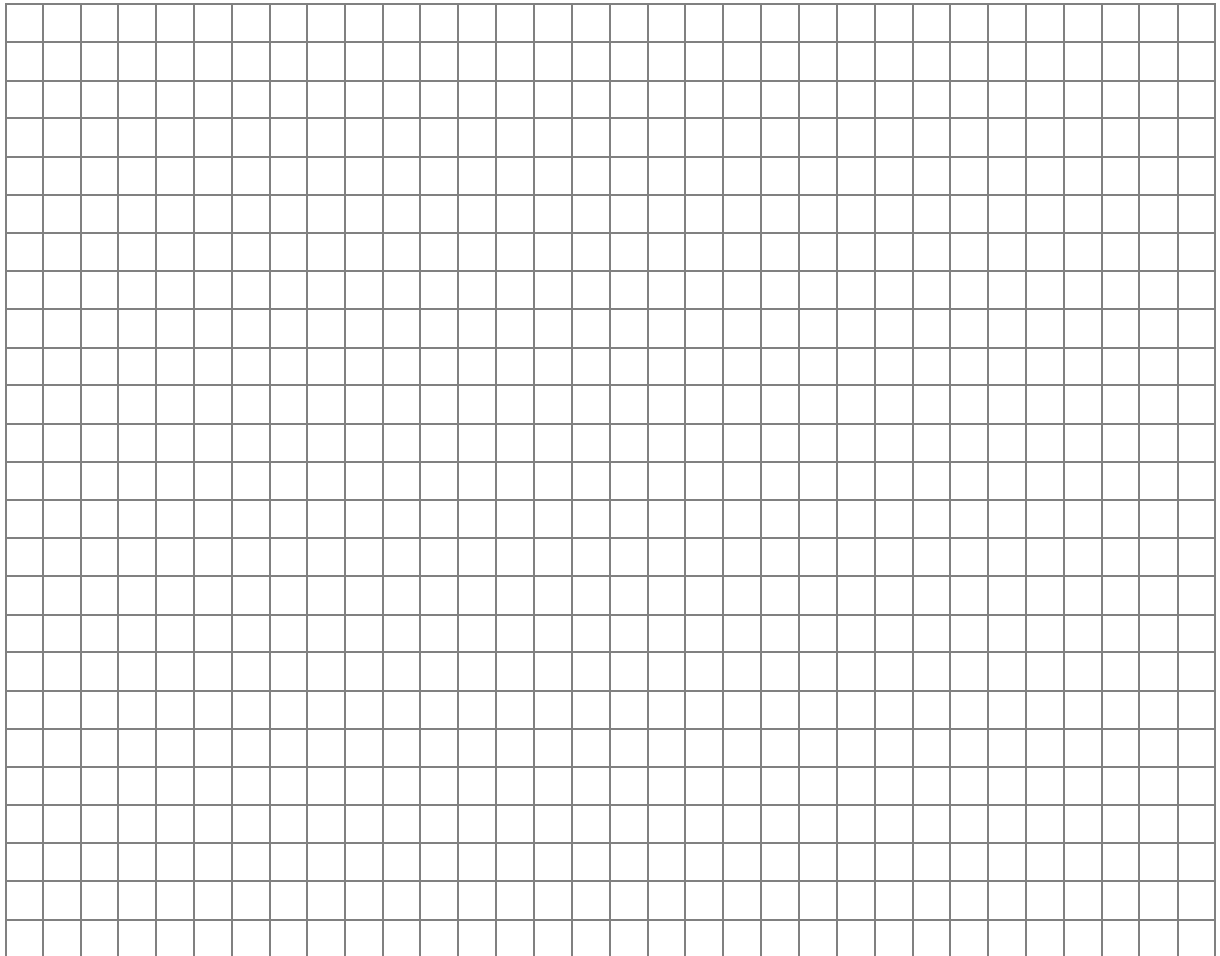
Wymagania ogólne	<i>V. Ocena zagrożeń i ograniczeń, docenianie społecznych aspektów rozwoju i zastosowań informatyki.</i>
Wymagania szczegółowe	<i>7. Wykorzystywanie komputera i technologii informacyjno-komunikacyjnych do rozwijania zainteresowań, opisywanie zastosowań informatyki, ocena zagrożeń i ograniczeń, aspekty społeczne rozwoju i zastosowań informatyki.</i> <i>Zdający:</i> <i>3) stosuje normy etyczne i prawne związane z rozpowszechnianiem programów komputerowych, bezpieczeństwem i ochroną danych oraz informacji w komputerze i w sieciach komputerowych.</i>
Rozwiązanie	PFFF
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Zadanie 12. (0–1)

Algorytmem sortowania przez porównania jest

	P	F
sortowanie przez wybór.		
sortowanie kubełkowe.		
sortowanie przez wstawianie.		
sortowanie szybkie.		

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>11) opisuje podstawowe algorytmy.</i>
Rozwiązanie	PFPP
Schemat punktowania	1 pkt – poprawne zaznaczenie wszystkich odpowiedzi. 0 pkt – błędne zaznaczenia lub ich brak.

Algorytm

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>2) stosuje podejście algorytmiczne do rozwiązywania problemu,</i> <i>4) dobiera efektywny algorytm do rozwiązania sytuacji problemowej i zapisuje go w wybranej notacji,</i> <i>5) posługuje się podstawowymi technikami algorytmicznymi,</i> <i>11) opisuje podstawowe algorytmy,</i> <i>12) projektuje rozwiązanie problemu (realizację algorytmu) i dobiera odpowiednią strukturę danych.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za każdą poprawną odpowiedź – 1 punkt.	3	7
b	Za poprawny algorytm – 4 punkty. W przypadku niepoprawnego algorytmu: – za prawidłowe wyznaczenie maksimum długości bloków jedynek – 1 punkt. – za prawidłowe wyznaczenie długości bloków jedynek – 1 punkt.	4	

Zadanie 13. Zapisy binarne (0–7) – rozwiązanie

W tym zadaniu badamy zapisy binarne dodatnich liczb całkowitych. Rozważamy następujący algorytm:

Specyfikacja

Dane: dodatnia liczba całkowita n

Wynik: dodatnia liczba całkowita j równa **liczbie** jedynek w zapisie binarnym liczby n

- (1) $j \leftarrow 0$;
- (2) **powtarzaj**
- (3) **jeśli** $n \bmod 2 = 1$, **to**
- (4) $j \leftarrow j + 1$;
- (5) $n \leftarrow n \div 2$;
- (6) **aż** $n = 0$;

Uwaga: użyte operatory mod i div oznaczają odpowiednio resztę z dzielenia i dzielenie całkowite, np. $5 \bmod 2 = 1$, $5 \div 2 = 2$, $6 \bmod 2 = 0$, $6 \div 2 = 3$.

- a) Przeanalizuj powyższy algorytm i podaj wartości zmiennej j po zakończeniu jego działania dla $n = 183$ oraz dla $n = 1022$. Uzupełnij brakujący fragment specyfikacji.

n	j
183	6
1022	9

Miejsce na obliczenia.

$$183 = (10110111)_2$$

$$1022 = (111111110)_2$$

- b) Ułóż algorytm i zapisz go w wybranej przez siebie notacji (lista kroków lub język programowania, który wybrałeś na egzamin), który dla danej dodatniej liczby całkowitej n oblicza maksymalną liczbę kolejnych jedynek pojawiających się w zapisie binarnym tej liczby.

Specyfikacja

Dane: dodatnia liczba całkowita n

Wynik: dodatnia liczba całkowita m – maksymalna liczba kolejnych jedynek w zapisie binarnym n

Przykład: dla $n = 187$ wynikiem jest $m = 3$, ponieważ $187 = (10111011)_2$

Algorytm

Każdy ciąg kolejnych jedynek w zapisie binarnym liczby, który nie można już wydłużyć, nazywamy blokiem. W zapisie binarnym liczby 187 mamy trzy bloki jedynek o długościach jeden, trzy i dwa: $(10111011)_2$. Naszym celem jest policzenie długości najdłuższego bloku. Modyfikujemy algorytm z podpunktu a), wyznaczając kolejne cyfry liczby n , od cyfr najmniej znaczących do cyfr najbardziej znaczących. W momencie wykrycia bloku (pierwszej jedynki w tym bloku) rozpoczynamy zliczanie jedynek w nim zawartych, aż w zapisie binarnym napotkamy zero lub wyznaczymy już wszystkie cyfry zapisu. Po przetworzeniu bloku porównujemy jego długość z długością dotychczas najdłuższego bloku i jeśli policzona długość jest większa od dotychczas największej, aktualizujemy informację o długości najdłuższego bloku.

Oto zapis opisanego słowami algorytmu:

- (1) $m \leftarrow 0$;
- (2) **powtarzaj**
 // m – długość dotychczas najdłuższego bloku
- (3) **jeśli** $n \bmod 2 = 1$, **to**
 // nowy blok
- (4) $dl_bloku \leftarrow 0$; // tu zliczamy liczbę jedynek w bloku
- (5) **powtarzaj**
- (6) $dl_bloku \leftarrow dl_bloku + 1$;
- (7) $n \leftarrow n \div 2$;
- (8) **aż** $n \bmod 2 = 0$;
- (9) **jeśli** $dl_bloku > m$, **to**
- (10) $m \leftarrow dl_bloku$;
- (11) $n \leftarrow n \div 2$;
- (12) **aż** $n = 0$;

Komentarz

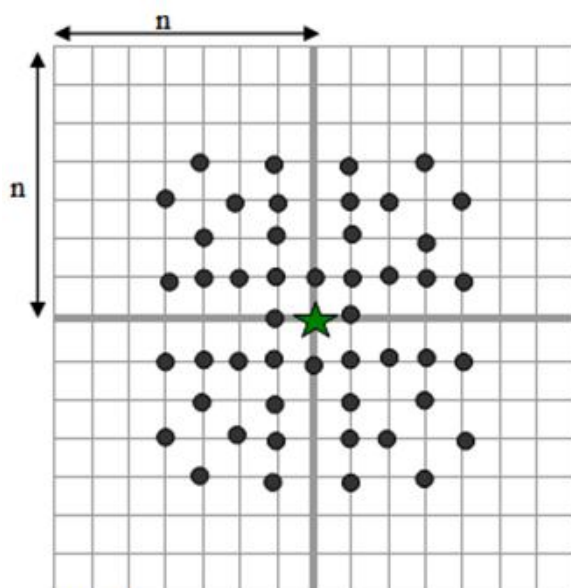
Podpunkt a) w tym zadaniu nie powinien sprawić żadnych trudności. Przedstawiony w nim algorytm jest typowym szkolnym algorytmem wyznaczania kolejnych cyfr dodatniej liczby całkowitej w jej zapisie binarnym, poczynając od cyfry najmniej znaczącej, a kończąc na cyfrze najbardziej znaczącej. Warto zauważyć, że w ten sam sposób można wyznaczyć cyfry w zapisie pozycyjnym przy dowolnej podstawie p , $2 \leq p \leq 10$. Wystarczy wykonywać

operacje dzielenia całkowitego i brania reszty z dzielenia z parametrem p zamiast 2. Dla liczby naturalnej n , $n \bmod p$ jest najmniej znaczącą cyfrą w zapisie pozycyjnym liczby n przy podstawie p . Dla przykładu $187 \bmod 10 = 7$, $187 \bmod 2 = 1$. Jeśli najmniej znaczącą cyfrą w zapisie przy podstawie p liczby n jest cyfra c , to $n = n' \cdot p + c$, dla pewnej liczby naturalnej n' . Wówczas $n \div p = n'$ i kolejną cyfrą w zapisie n jest najmniej znacząca cyfra w zapisie n' . Te własności właśnie wykorzystano w algorytmie z podpunktu a).

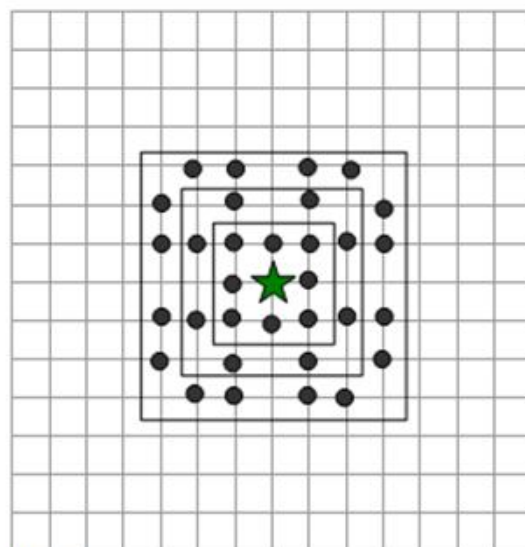
W punkcie b), oprócz wyznaczania cyfr liczby n , należy zliczać jedynek w blokach kolejnych jedynek. Tutaj najpierw trzeba wykryć blok. To jest proste – blok rozpoczyna się od jedynki. Następnie należy zliczać w pętli kolejne napotkane jedynek, aż pojawi się zero. Tak naprawdę w tym celu wykorzystujemy pętlę z algorytmu w punkcie a). Tak więc cały algorytm składa się z dwóch zagnieżdżonych pętli, których struktury są podobne do pętli z punktu a).

Można sobie wyobrazić inne rozwiązanie. W tym nowym rozwiązaniu zliczamy jedynek za każdym razem od momentu pojawienia się pierwszej jedynki w bloku. Pojawienie się zera powoduje wyzerowanie licznika jedynek. Oto formalny zapis tego algorytmu:

- (1) $m \leftarrow 0$;
- (2) $dl_bloku \leftarrow 0$;
- (3) **dopóki** $n \neq 0$ **wykonuj**
- (4) **jeśli** $n \bmod 2 = 1$ **to**
- (5) $dl_bloku \leftarrow dl_bloku + 1$;
- (6) **w przeciwnym wypadku**
- (7) **jeśli** $dl_bloku > m$ **to**
- (8) $m := dl_bloku$;
- (9) $dl_bloku \leftarrow 0$;
- (10) $n \leftarrow n \div 2$;

Zadanie 14. Fani (0–9)

Rys1



Rys2

Fani spotykają się z ulubionymi gwiazdami filmowymi w sali, której podłoga ma kształt kwadratu. Podłogę podzielono pionowymi i poziomymi liniami na mniejsze kwadraty rozmiaru 1×1 . Zaproszona gwiazda zawsze siada w środku sali. Fani zajmują miejsca w sali na przecięciach linii. Jedno miejsce może zająć co najwyżej jedna osoba.

Po wejściu na salę wypełniają oni kolejno strefy zaznaczone na Rys.2. kwadratami, każdą strefę maksymalnie, jak się da. Fani zajmują tylko te miejsca, z których gwiazda jest widoczna, co oznacza, że na odcinku łączącym ich miejsce z miejscem zajmowanym przez gwiazdę nie ma żadnej innej osoby. Na potrzeby zadania fanów i gwiazdę utożsamiamy z punktami, które zajmują. Na Rys. 1. przedstawione jest przykładowe rozmieszczenie wszystkich osób zgodnie z opisem. Na Rys. 2. pokazany jest przykładowe całkowite wypełnienie sali w 3 strefach. Jak widać w I strefie zasiadzie 8 fanów, w II strefie – 8, a w III aż 16.

14.1 Uzupełnij poniższą tabelę

Strefa	Maksymalna liczba fanów w kwadracie	Przyrost fanów w stosunku do poprzedniego poziomu
I	8	-
II	16	8
III	32	16
IV		
V		
VI		

14.2 Potraktuj salę (Rys. 1.) jako układ kartezjański, przyjmując pozycję gwiazdy jako punkt $(0,0)$, a pozycje fanów jako punkty kratowe o określonych współrzędnych (x,y) , odpowiedz na pytania:

a) W której strefie znajduje się pozycja o współrzędnych $(17;18)$?

.....

b) Podaj dwie pozycje, najbliższe położone względem pozycji $(4,3)$, zasłanianie przez fana stojącego na tej pozycji.

.....

c) Podaj współrzędne dwóch miejsc, których zajęcie przez fanów uniemożliwi oglądanie gwiazdy przez fana znajdującego się w polu $(8; 12)$.

.....

d) Czy fan może stanąć na każdej pozycji, której jedna ze współrzędnych jest równa 1?

.....

e) Czy fan może stanąć w miejscu o współrzędnych $(13; 39)$? Odpowiedź uzasadnij.

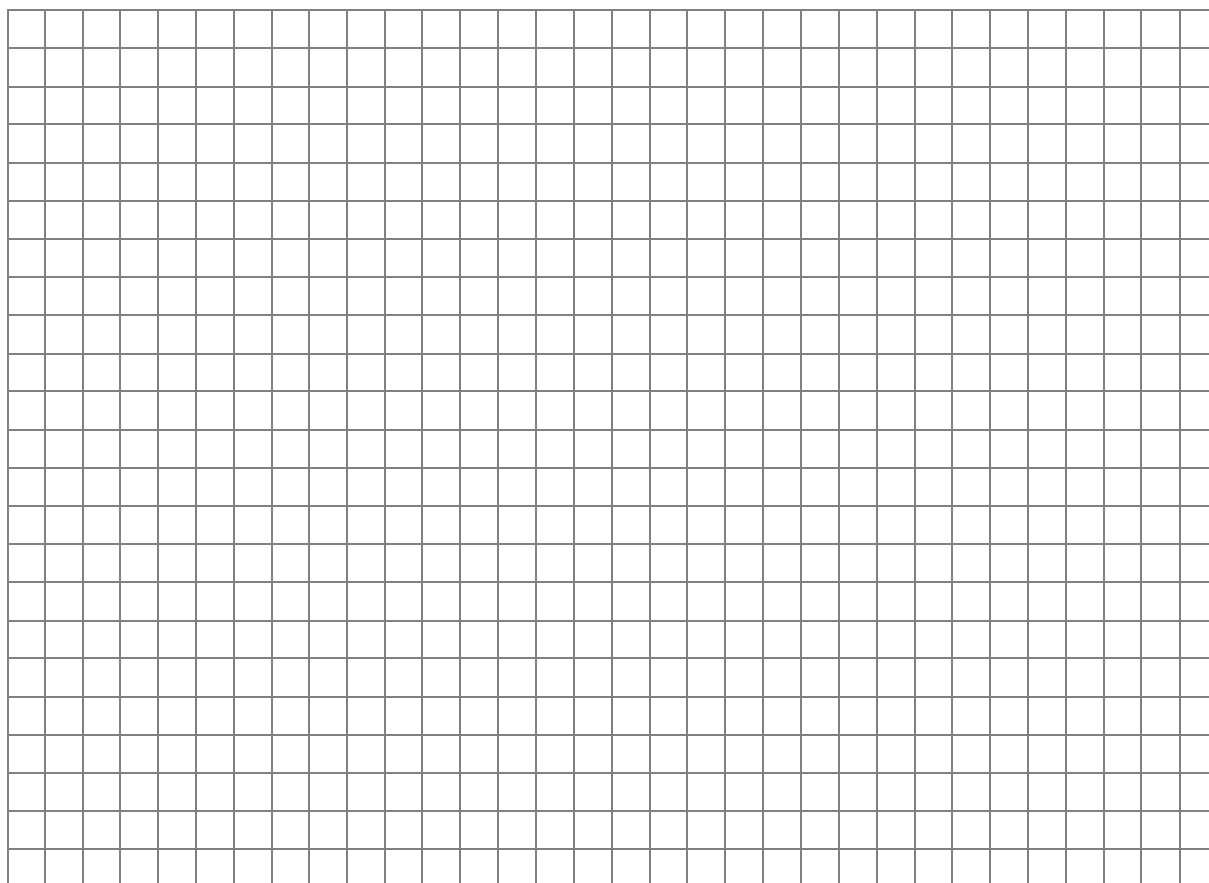
.....

14.3 Zapisz algorytm (w postaci listy kroków, schematu blokowego lub w wybranym języku programowania) sprawdzający, czy fan może stanąć w miejscu o danych współrzędnych (x,y) .

Specyfikacja

Dane: x,y – liczby całkowite określające położenie fana względem gwiazdy

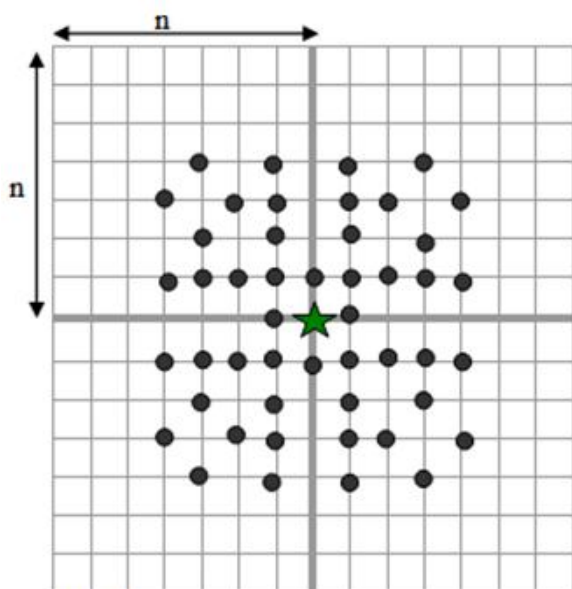
Wynik: Komunikat *TAK*, jeżeli fan widzi gwiazdę lub komunikat *NIE*, jeżeli fan gwiazdy nie widzi.

Algorytm

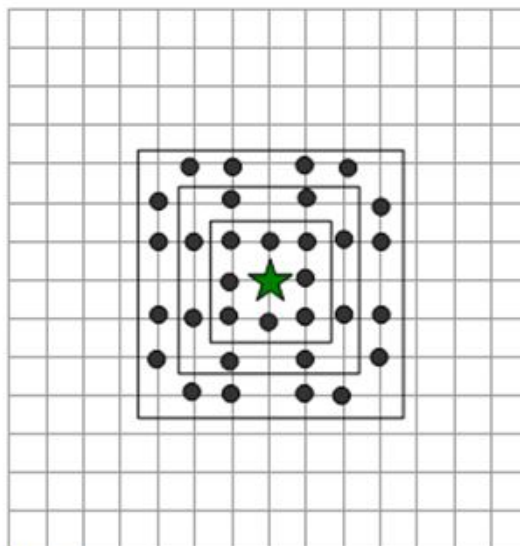
Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin,</i> <i>5) posługuje się podstawowymi technikami algorytmicznymi,</i> <i>6) ocenia własności rozwiązania algorytmicznego (komputerowego), np. zgodność ze specyfikacją, efektywność działania,</i> <i>11) opisuje podstawowe algorytmy.</i>

Schemat punktowania

Nr podpunktu	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
14.1.	Za poprawną uzupełnienie tabeli – 3 punkty. Za poprawnie uzupełnienie każdego wiersza – 1 punkt.	3	9
14.2.	Za poprawne podanie odpowiedzi do każdego podpunktu – 1 punkt; razem 4 punkty.	4	
14.3.	Za poprawny algorytm – 2 punkty. W przypadku algorytmu z błędnymi warunkami brzegowymi – 1 punkt.	2	

Zadanie 14. Fani (0–9)– rozwiązanie

Rys1



Rys2

Fani spotykają się z ulubionymi gwiazdami filmowymi w sali, której podłoga ma kształt kwadratu. Podłogę podzielono pionowymi i poziomymi liniami na mniejsze kwadraty rozmiaru 1x1. Zaproszona gwiazda zawsze siada w środku sali. Fani zajmują miejsca w sali na przecięciach linii. Jedno miejsce może zająć co najwyżej jedna osoba.

Po wejściu na salę wypełniają oni kolejno strefy zaznaczone na Rys. 2. kwadratami, każdą strefę maksymalnie, jak się da. Fani zajmują tylko te miejsca, z których gwiazda jest widoczna, co oznacza, że na odcinku łączącym ich miejsce z miejscem zajmowanym przez gwiazdę nie ma żadnej innej osoby. Na potrzeby zadania fanów i gwiazdę utożsamiamy z punktami, które zajmują. Na Rys. 1. przedstawione jest przykładowe rozmieszczenie wszystkich osób zgodnie z opisem. Na Rys. 2. pokazany jest przykładowe całkowite wypełnienie sali w 3 strefach. Jak widać w I strefie zasiądzie 8 fanów, w II strefie – 8, a w III aż 16.

14.1. Uzupełnij poniższą tabelę

Strefa	Maksymalna liczba fanów w kwadracie	Przyrost fanów w stosunku do poprzedniego poziomu
I	8	-
II	16	8
III	32	16
IV	48	16
V	80	32
VI	96	16

14.2. Traktując salę (Rys. 1.) jako układ kartezjański, przyjmując pozycję gwiazdy jako punkt (0,0), a pozycje fanów jako punkty kratowe o określonych współrzędnych (x,y) odpowiedź na pytania:

- W której strefie znajduje się pozycja o współrzędnych (17; -18)?
XVIII... (max z |x| lub |y|)
- Podaj dwie pozycje, najbliższe położone względem pozycji (4,3), zasłaniane przez fana stojącego na tej pozycji.
(8; 6) oraz (12; 9)
- Podaj współrzędne dwóch miejsc, których zajęcie przez fanów uniemożliwi oglądanie gwiazdy przez fana znajdującego się w polu (8; 12).
(2; 3) oraz (4; 6)
- Czy fan może stanąć w każdym punkcie, którego jedna ze współrzędnych jest równa 1?
TAK
- Czy fan może stanąć w miejscu o współrzędnych (13; 39)? Odpowiedź uzasadnij.
NIE. Współrzędne x i y fana muszą być liczbami względnie pierwszymi.

14.3. Zapisz algorytm (w postaci listy kroków, schematu blokowego lub w wybranym języku programowania) sprawdzający, czy fan może stanąć w miejscu o danych współrzędnych (x,y).

Specyfikacja

Dane: x,y – liczby całkowite określające położenie fana względem gwiazdy

Wynik: komunikat *TAK*, jeżeli fan widzi gwiazdę lub komunikat *NIE* – jeżeli fan gwiazdy nie widzi

Algorytm

Poniżej zamieszczony jest algorytm zapisany w języku programowania C:

```
int nwd(int a, int b)
{ while (a != b)
    if (a > b) a -= b; else b -= a;
  return a;
}
int main()
{
  cin >> x >> y;
```

```

if (x<0)x=-x; if(y<0)y=-y; //sprowadzenie problemu do liczb
//nieujemnych
if ((x==0)&&(y==0)) cout<<"NIE"; else
if ((x==0)&&(y!=1)|| (x!=1)&&(y==0)) cout<<"NIE"; else
if ((x==0)&&(y==1)|| ((y==0)&&(x==1))) cout<<"TAK";else
if (nwd(x,y)==1) cout<<"TAK"; else
cout<<"NIE";
}

```

Komentarz

Kluczem do poprawnego sformułowania algorytmu jest spostrzeżenie, że fani mogą zająć tylko te punkty kratowe o współrzędnych (x, y) , które nie zostaną przesłonięte przez żaden inny punkt kratowy. Inaczej, na odcinku, łączącym punkt $(0, 0)$ i punkt kratowy (x, y) , który ma zająć fan, nie może wystąpić żaden inny punkt o współrzędnych całkowitych. Punktami spełniającymi ten warunek są punkty, których $NWD(x, y)=1$, czyli takie, których współrzędne są liczbami względnie pierwszymi. Dlaczego? Gdyby współrzędne (x, y) nie były względnie pierwsze, to znaczyłoby, że na wymienionym odcinku musiałby się znaleźć punkt o współrzędnych całkowitych $(x / NWD(x, y), y / NWD(x, y))$, który zasłaniałby gwiazdę.

Dla uproszczenia algorytmu można sprowadzić warunki zadania do przypadku, gdy $x > 0$ i $y > 0$, ponieważ $NWD(x, y) = NWD(|x|, |y|)$. Jeżeli którakolwiek współrzędna jest równa zero, to można wykorzystać fakt, że $NWD(x, 0) = |x|$.

Punkt 14.1 oraz 14.2 z podpunktami od a) do e) pomagają w analizie problemu przed sformułowaniem i zapisaniem algorytmu. Po poprawnym uzasadnieniu odpowiedzi z punktu 14.2.e) zapisanie algorytmu nie powinno przysporzyć trudności zdającym.

Zadanie 15. Sortowanie (0–6)

W tym zadaniu rozważamy algorytmy sortujące niemalejąco n -elementową tablicę liczb całkowitych $a[1..n]$, gdzie n jest dodatnią liczbą całkowitą. Algorytm sortowania nazywamy lokalnym, gdy podczas sortowania można porównywać i zamieniać ze sobą tylko sąsiednie elementy tablicy. Na przykład dopuszczalne jest porównanie i zamiana elementów $a[5]$ i $a[6]$, natomiast nie można bezpośrednio porównywać i zamieniać ze sobą elementów $a[5]$ i $a[7]$.

- a) Które z następujących algorytmów sortowania są algorytmami lokalnymi: **bąbelkowy**, **przez wstawianie liniowe**, **szybki**? Udziel odpowiedzi wpisując słowa TAK lub NIE w prawej kolumnie tabeli poniżej.

Algorytm	Czy jest lokalny?
Bąbelkowy	
Przez wstawianie liniowe	
Szybki	

- b) Dla tablicy $a[1..4] = [3, 2, 4, 1]$ algorytm sortowania przez wstawianie liniowe wykona dokładnie 4 zamiany sąsiednich elementów: (3 z 2), (4 z 1), (3 z 1), (2 z 1). Uzupełnij luki w podanych poniżej tablicach różnymi liczbami całkowitymi tak, aby algorytm sortowania przez wstawianie liniowe wykonał na każdej z nich dokładnie 11 zamian sąsiednich elementów.

Tablica 1.

Pozycja	1	2	3	4	5	6	7	8	9	10
Zawartość	10	1	2		4	5	6		7	8

Tablica 2.

Pozycja	1	2	3	4	5	6	7	8	9	10
Zawartość	1	2	3	5	4					

- c) Załóżmy teraz, że w jednym kroku możemy posortować blok kolejnych elementów tablicy dłuższy niż 2. Na przykład gdybyśmy mogli sortować bloki o długościach do 9 elementów, wówczas tablicę 10-elementową można by posortować w trzech krokach: najpierw w jednym kroku sortujemy ostatnie 9 elementów. W następnym kroku sortujemy pierwsze 9 elementów. Teraz wiemy, że element najmniejszy jest już na swojej, czyli pierwszej, pozycji w tablicy. Jeszcze jedno sortowanie ostatnich 9 elementów kończy sortowanie całego ciągu. Oznaczmy przez $Sort(i, j)$ sortowanie w jednym kroku bloku kolejnych elementów z pozycji od i do j . Wówczas powyższe sortowanie można zapisać w następujący sposób:

- (1) $Sort(2, 10)$;
- (2) $Sort(1, 9)$;
- (3) $Sort(2, 10)$;

Przykład:

Początkowa zawartość tablicy:

$$a = [10, 2, 8, 4, 6, 5, 7, 9, 3, 1]$$

Sortowanie ostatnich 9 elementów ($Sort(2,10)$):

$$a = [10, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

Sortowanie pierwszych 9 elementów ($Sort(1,9)$):

$$a = [1, 2, 3, 4, 5, 6, 7, 8, 10, 9]$$

Sortowanie ostatnich 9 elementów ($Sort(2,10)$):

$$a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].$$

Zapisz algorytm sortowania tablicy **1000**-elementowej w co najwyżej 6 krokach, przy założeniu, że w jednym kroku można posortować blok złożony z co najwyżej **500** elementów.

Kolejne kroki algorytmu:

1.	
2.	
3.	
4.	
5.	
6.	

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>2) stosuje podejście algorytmiczne do rozwiązywania problemu,</i> <i>5) posługuje się podstawowymi technikami algorytmicznymi,</i> <i>11) opisuje podstawowe algorytmy,</i> <i>16) opisuje własności algorytmów na podstawie ich analizy.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za poprawne uzupełnienie wszystkich wierszy tabeli – 1 punkt.	1	6
b	Za poprawne uzupełnienie pierwszej tabeli (pozycja 4 – 3, pozycja 8 – 9) – 1 punkt. Za poprawne uzupełnienie drugiej tabeli (10,9,8,7,6) – 1 punkt.	2	
c	Za poprawny algorytm – 3 punkty. W przypadku błędnego algorytmu: ustawienie minimum na pierwszej pozycji – 1 punkt. ustawienie maksimum na ostatniej pozycji – 1 punkt.	3	

Zadanie 15. Sortowanie (0–6) – rozwiązanie

W tym zadaniu rozważamy algorytmy sortujące niemalejąco n -elementową tablicę liczb całkowitych $a[1..n]$, gdzie n jest dodatnią liczbą całkowitą. Algorytm sortowania nazywamy lokalnym, gdy podczas sortowania można porównywać i zamieniać ze sobą tylko sąsiednie elementy tablicy.

Na przykład dopuszczalne jest porównanie i zamiana elementów $a[5]$ i $a[6]$, natomiast nie można bezpośrednio porównywać i zamieniać ze sobą elementów $a[5]$ i $a[7]$.

- a) Które z następujących algorytmów sortowania są algorytmami lokalnymi: **bąbelkowy**, **przez wstawianie liniowe**, **szybki**? Udziel odpowiedzi wpisując słowa TAK lub NIE w prawej kolumnie tabeli poniżej:

Algorytm	Czy jest lokalny?
Bąbelkowy	TAK
Przez wstawianie liniowe	TAK
Szybki	NIE

- b) Dla tablicy $a[1..4] = [3,2,4,1]$ algorytm sortowania przez wstawianie liniowe wykona dokładnie 4 zamiany sąsiednich elementów. Uzupełnij luki w podanych poniżej tablicach różnymi liczbami całkowitymi tak, aby algorytm sortowania przez wstawianie liniowe wykonał na każdej z nich dokładnie 11 zamian sąsiednich elementów.

Tablica 1.

Pozycja	1	2	3	4	5	6	7	8	9	10
Zawartość	10	1	2	3	4	5	6	9	7	8

Tablica 2.

Pozycja	1	2	3	4	5	6	7	8	9	10
Zawartość	1	2	3	5	4	10	9	8	7	6

- c) Załóżmy teraz, że w jednym kroku możemy posortować blok kolejnych elementów tablicy dłuższy niż 2. Na przykład, gdybyśmy mogli sortować bloki o długościach do 9 elementów, wówczas tablicę 10-elementową można by posortować w trzech krokach:

najpierw w jednym kroku sortujemy ostatnie 9 elementów. W następnym kroku sortujemy pierwsze 9 elementów. Teraz wiemy, że element najmniejszy jest już na swojej, czyli pierwszej pozycji w tablicy. Jeszcze jedno sortowanie ostatnich 9 elementów kończy sortowanie całego ciągu. Oznaczmy przez $Sort(i,j)$ sortowanie w jednym kroku bloku kolejnych elementów z pozycji od i do j . Wówczas powyższe sortowanie można zapisać w następujący sposób:

(4) $Sort(2,10)$;

(5) $Sort(1,9)$;

(6) $Sort(2,10)$;

Przykład:

Początkowa zawartość tablicy:

$a = [10, 2, 8, 4, 6, 5, 7, 9, 3, 1]$

Sortowanie ostatnich 9 elementów ($Sort(2,10)$):

$a = [10, 1, 2, 3, 4, 5, 6, 7, 8, 9]$

Sortowanie pierwszych 9 elementów ($Sort(1,9)$):

$a = [1, 2, 3, 4, 5, 6, 7, 8, 10, 9]$

Sortowanie ostatnich 9 elementów ($Sort(2,10)$):

$a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$.

Zapisz algorytm sortowania tablicy **1000**-elementowej w co najwyżej 6 krokach, przy założeniu, że w jednym kroku można posortować blok złożony z co najwyżej **500** elementów.

Kolejne kroki algorytmu:

1.	$Sort(1,500)$
2.	$Sort(251,750)$
3.	$Sort(501,1000)$
4.	$Sort(1,500)$
5.	$Sort(251,750)$
6.	$Sort(1,500)$

Komentarz

To zadanie sprawdza rozumienie zasad działania podstawowych algorytmów sortowania. Wszystkie trzy algorytmy wymienione w punkcie a), to algorytmy sortujące polegające na porównywaniu wartości elementów znajdujących się na różnych pozycjach w tablicy a i zamianie ich miejscami, jeśli element o większej wartości poprzedza w tablicy element o mniejszej wartości. W algorytmach sortowania, bąbelkowym i przez wstawianie liniowe, porównywane i zamieniane są tylko pary sąsiednich elementów w tablicy. Tak nie jest w algorytmie szybkim, ponieważ w przeciwnym razie stracilibyśmy walor „szybkości”. Żeby rozwiązać punkt b) zastanówmy się, ile zamian wykonamy sortując algorytmem przez wstawianie tablicę $a[1..n]$ o zadanej zawartości. Oznaczmy przez $b[i]$ liczbę elementów w tablicy a znajdujących się na pozycjach o indeksach mniejszych od i , ale o wartościach większych od wartości elementu $a[i]$. Nietrudno zauważyć, że podczas wstawiania elementu $a[i]$ do uporządkowanego już fragmentu tablicy $a[1..i-1]$, zostanie on zamieniony ze wszystkimi elementami o wartościach większych, a jest ich dokładnie $b[i]$. Zatem łączna liczba zamian wykonywanych w algorytmie sortowania przez wstawianie tablicy a wyniesie $b[1] + b[2] + \dots + b[n]$. Poniżej pokazujemy zawartość tablicy b dla przykładów z punktu b).

Tablica 1.

<i>pozycja</i>	1	2	3	4	5	6	7	8	9	10
<i>Zawartość</i>	10	1	2	3	4	5	6	9	7	8
<i>tablica b</i>	0	1	1	1	1	1	1	1	2	2

Tablica 2.

<i>pozycja</i>	1	2	3	4	5	6	7	8	9	10
<i>zawartość</i>	1	2	3	5	4	10	9	8	7	6
<i>tablica b</i>	0	0	0	0	1	0	1	2	3	4

Do rozwiązania punktu c) możemy zaadoptować algorytm sortowania bąbelkowego. Gdyby tablica *a* liczyła tylko cztery elementy, to do jej posortowania wystarczy (i potrzeba) 6 wywołań procedury *Sort*:

Sort(1,2), *Sort*(2,3), *Sort*(3,4), *Sort*(1,2), *Sort*(2,3), *Sort*(1,2).

Po trzech pierwszych wywołaniach *Sort* element o największej wartości znajdzie się już na swojej docelowej pozycji nr 4. Dwa następne wywołania zagwarantują, że na pozycji nr 3 w *a* znajdzie się element drugi, licząc od największego. Ostatnie wywołanie *Sort* porządkuje dwa najmniejsze elementy i ustawia je na właściwych, dwóch pierwszych pozycjach w tablicy *a*. Podobnie dzieje się w zaproponowanym rozwiązaniu sortowania tablicy 1000 elementowej. Pierwsze trzy wywołania *Sort* gwarantują umieszczenie 250 największych elementów na ich docelowych pozycjach. Kolejne dwa wywołania *Sort* umieszczają w dobrym porządku, na docelowych pozycjach od 501 do 750, kolejne 250 elementów. Ostatnie sortowanie porządkuje pierwszych 500 najmniejszych elementów. Alternatywne rozwiązanie mogłoby polegać na zaadaptowaniu sortowania przez wstawianie, kiedy to wstawiamy bloki po 250 elementów:

Sort(1,500), *Sort*(251,750), *Sort*(1,500), *Sort*(501,1000), *Sort*(251,750), *Sort*(1,500).

Zadanie 16. Miejsce zerowe (0–6)

Przedstawiona poniżej rekurencyjna funkcja $Mzer$ znajduje metodą bisekcji miejsce zerowe funkcji f , ciągłej w przedziale $\langle a, b \rangle$, z dokładnością do $\frac{\varepsilon}{2}$ [epsilon].

Specyfikacja

Dane: liczby a i b takie, że $a < b$ oraz $f(a) \cdot f(b) < 0$

liczba $\varepsilon > 0$

Wynik: liczba rzeczywista x z przedziału $\langle a, b \rangle$ taka, że $f(x + \alpha) = 0$ dla pewnego α takiego,

$$\text{że } -\frac{\varepsilon}{2} \leq \alpha \leq \frac{\varepsilon}{2}$$

Funkcja $Mzer(a, b, \varepsilon)$

1. jeżeli $b - a > \varepsilon$, to wykonaj:

a. $s \leftarrow (a + b) / 2$

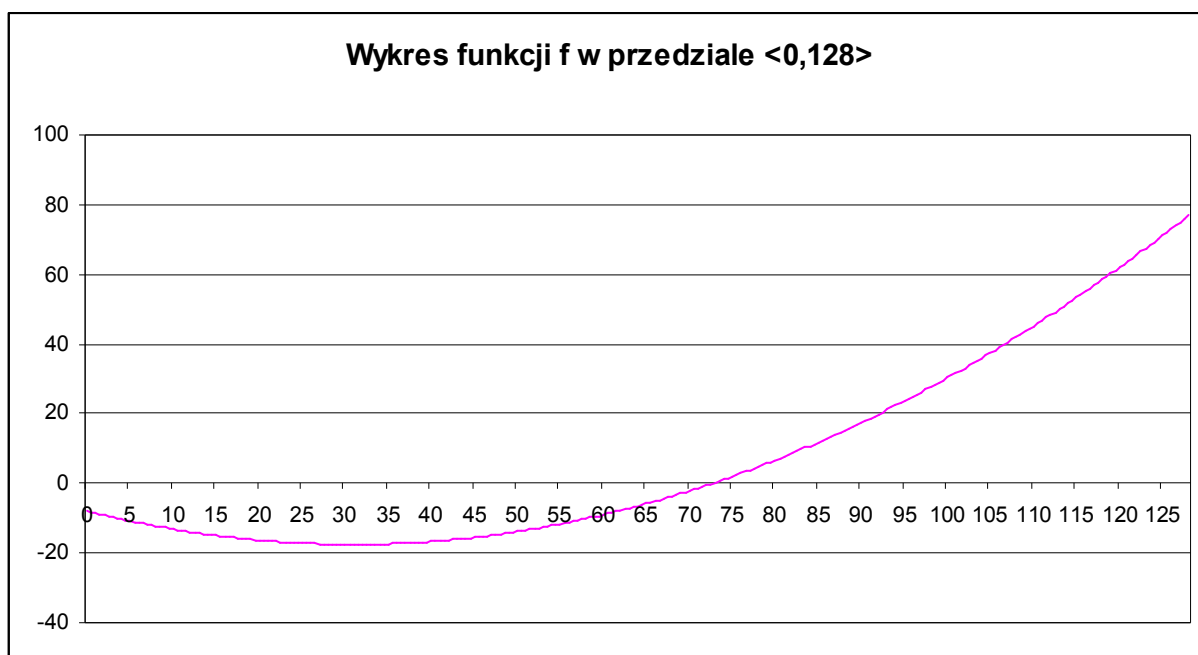
b. jeżeli $f(s) = 0$, to podaj s jako wynik

c. jeżeli $f(a) \cdot f(s) < 0$, to podaj $Mzer(a, s, \varepsilon)$ jako wynik

w przeciwnym przypadku podaj $Mzer(s, b, \varepsilon)$ jako wynik

2. w przeciwnym przypadku podaj $(a + b) / 2$ jako wynik

a) Poniżej prezentujemy fragment wykresu funkcji f , dla której wywołujemy funkcję $Mzer$:



Wykres przecina oś OX w punkcie $72,7$. Załóżmy, że funkcja M_{zer} została wywołana dla $a = 0$ i $b = 128$. W poniższej tabelce podaj liczbę kolejnych rekurencyjnych wywołań funkcji M_{zer} przy podanych poniżej początkowych wartościach ε .

ε	liczba wywołań M_{zer}
10	4
32	
25	
5	
$\frac{1}{5}$	

Miejsce na obliczenia

[illegible]

- b) Poniżej prezentujemy zapis algorytmu opisanego funkcją $Mzer$ w postaci nierekurencyjnej. Zapis poniższego algorytmu jest niepełny, uzupełnij brakujące elementy tak, aby realizował tę samą metodę poszukiwania miejsca zerowego, którą opisuje funkcja $Mzer$.

Algorytm:

1. dopóki $b - a > \varepsilon$ wykonuj
 - a) $s \leftarrow \dots\dots\dots$
 - b) jeżeli $f(s) = 0$, to podaj s jako wynik i zakończ wykonywanie algorytmu
 - c) jeżeli $f(a) \cdot f(s) < 0$, to $b \leftarrow \dots\dots\dots$
w przeciwnym przypadku $a \leftarrow s$
2. podaj $(a+b)/2$ jako wynik

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>2) stosuje podejście algorytmiczne do rozwiązywania problemu,</i> <i>5) posługuje się podstawowymi technikami algorytmicznymi,</i> <i>9) stosuje rekurencję w prostych sytuacjach problemowych,</i> <i>11) opisuje podstawowe algorytmy.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za wszystkie poprawne odpowiedzi – 3 punkty. Za trzy poprawne odpowiedzi – 2 punkty. Za dwie poprawne odpowiedzi – 1 punkt.	3	6
b	Za każde poprawne uzupełnienie brakującego elementu – 1 punkt.	3	

Zadanie 16. Miejsce zerowe (0–6) – rozwiązanie

Przedstawiona poniżej rekurencyjna funkcja $Mzer$ znajduje metodą bisekcji **miejsce zerowe**

funkcji f ciągłej w przedziale $\langle a, b \rangle$ z dokładnością do $\frac{\varepsilon}{2}$ [epsilon].

Specyfikacja:

Dane: liczby a i b takie, że $a < b$ oraz $f(a) \cdot f(b) < 0$

liczba $\varepsilon > 0$

Wynik: liczba rzeczywista x z przedziału $\langle a, b \rangle$ taka, że $f(x + \alpha) = 0$ dla pewnego α

takiego, że $-\frac{\varepsilon}{2} \leq \alpha \leq \frac{\varepsilon}{2}$

Funkcja $Mzer(a, b, \varepsilon)$

1. jeżeli $b - a > \varepsilon$, to wykonaj:

a. $s \leftarrow \frac{a+b}{2}$

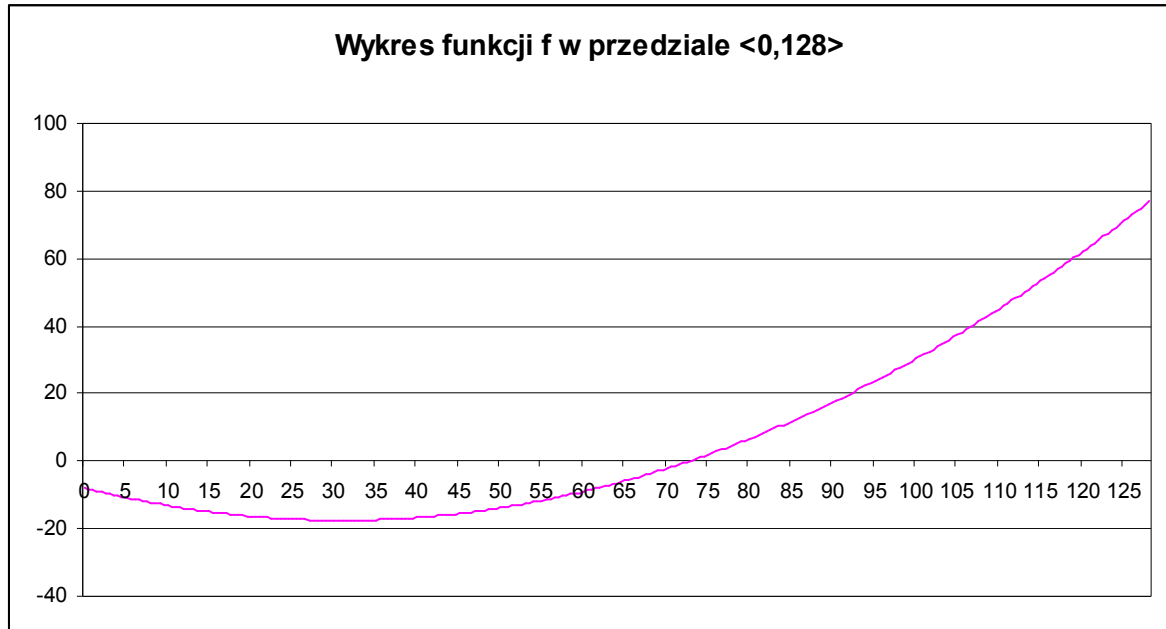
b. jeżeli $f(s) = 0$, to podaj s jako wynik

c. jeżeli $f(a) \cdot f(s) < 0$, to podaj $Mzer(a, s, \varepsilon)$ jako wynik

w przeciwnym przypadku podaj $Mzer(s, b, \varepsilon)$ jako wynik

2. w przeciwnym przypadku podaj $(a+b)/2$ jako wynik

Poniżej prezentujemy fragment wykresu funkcji f , dla której wywołujemy funkcję $Mzer$:



Wykres przecina oś OX w punkcie $72,7$. Załóżmy, że funkcja $Mzer$ została wywołana dla $a = 0$ i $b = 128$. W poniższej tabelce podaj liczbę kolejnych rekurencyjnych wywołań funkcji $Mzer$ przy podanych poniżej wartościach ε

ε	liczba wywołań $Mzer$
10	4
32	2
25	3
5	5
$\frac{1}{5}$	10

- b) Poniżej prezentujemy zapis algorytmu opisanego funkcją $Mzer$ w postaci nierekurencyjnej. Zapis poniższego algorytmu jest niepełny, uzupełnij brakujące elementy tak, aby realizował tę samą metodę poszukiwania miejsca zerowego, którą opisuje funkcja $Mzer$.

Algorytm

1. dopóki $b - a > \varepsilon$ wykonuj

a) $s \leftarrow \frac{a+b}{2}$

- b) jeżeli $f(s) = 0$, to podaj s jako wynik i zakończ wykonywanie algorytmu

- c) jeżeli $f(a) \cdot f(s) < 0$, to $b \leftarrow s$
w przeciwnym przypadku $a \leftarrow s$

2. podaj $(a+b)/2$ jako wynik

Komentarz

Podpunkt a

Aby obliczyć liczbę kolejnych rekurencyjnych wywołań funkcji $Mzer$ dla każdej podanej wartości ε , należy ustalić wartości zmiennych z jakimi będzie ona wywoływana w kolejnych krokach. Funkcja zakończy działanie w momencie, gdy $b - a < \varepsilon$.

Dla każdej dokładności startujemy od wywołania funkcji $Mzer(0, 128, \varepsilon)$, zaś kolejne wywołania to:

Dla $\varepsilon = 32$ mamy dwa wywołania:

$Mzer(64, 128, 32)$

$Mzer(64, 96, 32)$

Dla $\varepsilon = 25$ mamy trzy wywołania:

$Mzer(64, 128, 25)$

$Mzer(64, 96, 25)$

$Mzer(64, 80, 25)$

Dla $\varepsilon = 5$ mamy pięć wywołań:

$Mzer(64, 128, 5)$

$Mzer(64, 96, 5)$

$Mzer(64, 80, 5)$

$Mzer(72, 80, 5)$

$Mzer(72, 76, 5)$

Dla $\varepsilon = \frac{1}{5}$ mamy dziesięć wywołań:

$Mzer\left(64, 128, \frac{1}{5}\right)$

$Mzer\left(64, 96, \frac{1}{5}\right)$

$Mzer\left(64, 80, \frac{1}{5}\right)$

$Mzer\left(72, 80, \frac{1}{5}\right)$

$Mzer\left(72, 76, \frac{1}{5}\right)$

$Mzer\left(72, 74, \frac{1}{5}\right)$

$$\begin{aligned}
 &Mzer\left(72, 73, \frac{1}{5}\right) \\
 &Mzer\left(72, 5, 73, \frac{1}{5}\right) \\
 &Mzer\left(72, 5, 72, 75, \frac{1}{5}\right) \\
 &Mzer\left(72, 625, 72, 75, \frac{1}{5}\right)
 \end{aligned}$$

Podpunkt b

Aby uzupełnić luki w przedstawionej nierekurencyjnej wersji funkcji *Mzer*, należy zauważyć prostą własność: w kolejnych krokach zawsze ustalamy środek aktualnego przedziału, tzn.

$s \leftarrow \frac{a+b}{2}$, a następnie sprawdzamy czy miejsce zerowe funkcji leży na lewo od punktu s

(wtedy $b \leftarrow s$), czy na prawo od punktu s ($a \leftarrow s$).

Podpunkt a) sprowadza się do przeanalizowania algorytmu zaprezentowanego w treści zadania na konkretnych danych i wyznaczenia liczby wywołań funkcji rekurencyjnej. Funkcja *Mzer* opisuje podręcznikowy algorytm znajdowania miejsca zerowego funkcji ciągłej *f* metodą bisekcji. Zadaniem zaprezentowanej implementacji tego algorytmu jest podanie

miejsca zerowego funkcji w przedziale $[a, b]$, z dokładnością do $\frac{\varepsilon}{2}$ przy założeniu, że

$f(a) \cdot f(b) < 0$ (zauważmy, że ciągłość funkcji *f* w powiązaniu z warunkiem $f(a) \cdot f(b) < 0$ gwarantuje, że *f* ma miejsce zerowe w przedziale $[a, b]$).

W treści zadania przedstawiono pseudokod rekurencyjnej wersji algorytmu, która rozpoczyna się od sprawdzenia czy odległość między krańcami przedziału $[a, b]$ jest większa od ε (czyli podwojonej dokładności wyniku):

1. Gdy $b - a \leq \varepsilon$, wówczas mamy gwarancję że środek przedziału $[a, b]$ znajduje się nie dalej niż $\frac{\varepsilon}{2}$ od miejsca zerowego funkcji *f*. Dlatego też jako wynik zwracany jest właśnie środek przedziału, czyli wartość $(a+b)/2$.
2. Gdy $b - a > \varepsilon$, mamy dwie możliwości. Jeśli środek przedziału $[a, b]$ jest miejscem zerowym funkcji *f*, zwracamy go oczywiście jako wartość. W przeciwnym razie redukujemy zadanie wyszukania miejsca zerowego w $[a, b]$ do zadania poszukiwania miejsca zerowego w $[a, s]$ lub $[s, b]$, gdzie s to środek przedziału $[a, b]$. Efektem redukcji jest więc dwukrotne zmniejszenie długości przedziału, w którym poszukujemy miejsca zerowego.

Zauważmy, że gdyby wywołania funkcji *Mzer* dla $a=0$ i $b=128$ oraz podanych w punkcie a) wartości ε nigdy nie kończyły się znalezieniem dokładnej wartości miejsca zerowego (czyli spełnieniem warunku $f(s)=0$), odpowiedzi w punkcie a) sprowadzałyby się do wskazania ilokrotnie trzeba „połować” przedział o długości 128 aby uzyskać przedział o długości nie większej niż ε . W rozwiązaniu można by więc ograniczyć się do policzenia, ile takich „połowień” należy wykonać. Aby jednak mieć pewność poprawności rozwiązania, trzeba sprawdzić że wartość środka przedziału s rzeczywiście w naszym przykładzie nie „trafi” idealnie w miejsce zerowe (równe 72,7) w kolejnych wywołaniach rekurencyjnych. Taką skrupulatną analizę przedstawiliśmy omawiając rozwiązanie punktu a).

Punkt b) zadania wymaga zastosowania standardowej techniki zamiany rekurencji na iterację. Zamiast wywoływać funkcję z nowymi wartościami krańców przedziałów, zmieniamy w pętli wartości zmiennych a i b tak, aby odpowiadały one końcom coraz mniejszych przedziałów dla których wywoływana jest funkcja rekurencyjna $Mzer$. Przedstawiony szkielet algorytmu z pozostawionymi miejscami do uzupełnienia sugeruje sposób, w jaki w tym przypadku należy dokonać zamiany rekurencji na iterację.

Zadanie 17. Bruker (0–11)

Firma Bruker wygrała w przetargu kontrakt na ułożenie kostki na rynku starego miasta. Docelowo Bruker miała ułożyć $16\,500\text{ m}^2$ kostki obu rodzajów. Firma Bruker rozpoczęła prace w dniu **1.03.2013** roku i planowała je zakończyć w drugiej połowie listopada 2013 roku. W dniu 1.03.2013 roku, przed rozpoczęciem pracy, na rynku zgromadzono 500 m^2 kostki granitowej i 200 m^2 kostki bazaltowej.

W firmie zatrudniono **20 pracowników**, każdy z nich dziennie potrafi ułożyć $4,5\text{ m}^2$ powierzchni granitowej lub $3,8\text{ m}^2$ powierzchni bazaltowej. Firma pracuje 5 dni w tygodniu oprócz sobót i niedziel³. Bruker posiada samochody ciężarowe, każdy z nich jednorazowo może dostarczyć 32 m^2 kostki granitowej lub 28 m^2 kostki bazaltowej.

Kierownik opracował następujący system pracy: wszyscy pracownicy (20 osób) układają kostkę granitową do dnia, w którym rano zapas kostki granitowej jest mniejszy, niż ten zużywany codziennie przez cały zespół 20 pracowników. Wówczas połowa pracowników zostaje oddelegowana do pracy z kostką bazaltową, a połowa układa nadal kostkę granitową. Kiedy zapas kostki granitowej zostanie uzupełniony (rano zapas kostki wystarczy na cały dzień pracy dla wszystkich 20 pracowników) wszyscy ponownie układają tylko kostkę granitową.

Uzupełnianie zapasów następuje **wieczorem**, po pracy brukarzy, wg opisanej reguły: jeżeli po pracy zapas kostki granitowej jest mniejszy niż 40 m^2 , to przyjeżdżają **3** samochody z dostawą, jeżeli zapas wynosi od 40 m^2 do 100 m^2 włącznie – przyjeżdża **1** samochód, jeżeli zapas kostki jest większy niż 100 m^2 , wówczas nie ma dostawy. Zapas kostki bazaltowej jest uzupełniany w każdy **poniedziałek i środę** z użyciem jednego samochodu.

Korzystając z dostępnych narzędzi informatycznych, wykonaj poniższe polecenia. Odpowiedzi do podpunktów a), b), c) i e) zapisz w pliku `wyniki.txt`, a każdą z nich poprzedź literą oznaczającą ten podpunkt.

Uwaga: Pamiętaj, że firma Bruker pracuje 5 dni w tygodniu oprócz sobót i niedziel. Dostawy kostki odbywają się również tylko w dni robocze, zatem w swoich obliczeniach pominij soboty i niedziele.

- Podaj liczbę dostaw kostki bazaltowej, przy założeniu, że prace trwały do końca listopada.
- Podaj datę pierwszej dostawy kostki granitowej.
- Dla każdego pierwszego **roboczego** dnia miesiąca w okresie od 1.03.2013 do 1.11.2013 utwórz zestawienie złożone z daty i liczby metrów kwadratowych kostki granitowej oraz liczby metrów kwadratowych kostki bazaltowej ułożonych **do dnia wskazanego datą włącznie** (stan wieczorny).
- Utwórz wykres liniowy obrazujący poranny zapas kostki granitowej i kostki bazaltowej dla danych z okresu od 1.03.2013 do 1.11.2013 włącznie.
- Sprawdź, czy firmie Bruker uda się zakończyć pracę w planowanym terminie. Jeżeli tak, to podaj dzień zakończenia prac. Jeżeli nie, podaj powierzchnię ułożonej kostki w ostatnim dniu roboczym listopada 2013 roku.

Do oceny oddajesz plik(i) o nazwie(ach), zawierający(e)

tu wpisz nazwę(y) pliku (ów)

komputerową(e) realizację(e) Twoich obliczeń, plik tekstowy `wyniki.txt` zawierający odpowiedzi do podpunktów a), b), c) i e) oraz plik

tu wpisz nazwę pliku

zawierający reprezentację graficzną rozwiązania podpunktu d) zadania.

³ Dla uproszczenia symulacji pominij występowanie świąt państwowych i kościelnych.

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin,</i> <i>12) projektuje rozwiązanie problemu (realizację algorytmu) i dobiera odpowiednią strukturę danych,</i> <i>25) dobiera właściwy program użytkowy lub samodzielnie napisany program do rozwiązania zadania.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za poprawną odpowiedź – 2 punkty.	2	11
b	Za poprawną odpowiedź – 2 punkty.	2	
c	Za poprawną odpowiedź – 3 punkty. W przypadku jednego błędnego wiersza – 2 punkty. W przypadku dwóch błędnych wierszy – 1 punkt.	3	
d	Za poprawną odpowiedź – 2 punkty, w tym za: • typ wykresu i zakres danych, • pełny opis wykresu (tytuł, z wykresu można odczytać stan zapasów kostki granitowej i kostki bazaltowej w wybranym dniu).	2	
e	Za poprawną odpowiedź – 2 punkty.	2	

Zadanie 17. Bruker (0–11) – rozwiązanie

Plik zawierający komputerową realizację obliczeń *bruker.xlsx* oraz plik tekstowy zawierający odpowiedzi *wyniki.txt* znajdują się w folderze *BRUKER*.

Komentarz

Zadanie Bruker należy do typowych zadań symulacyjnych. Najtrudniejszą częścią zadania jest prawidłowe zasymulowanie cyklicznie powtarzających się wydarzeń, które są ze sobą powiązane. Rozwiązanie rozpoczynamy od wypisania dni, w których będą odbywały się prace, czyli wszystkich dni roboczych, począwszy od 1 marca 2013 roku do dnia 30 listopada 2013 roku oraz utworzenia kolumny z nazwą (numerem) dnia tygodnia.

W ciągu całego dnia zachodzi wiele wydarzeń, które są od siebie zależne. Żeby nie doszło do pomyłki, dobrze jest rozpisać wydarzenia dnia: stan początkowy (poranny) kostek obu rodzajów, liczbę ułożonych metrów kwadratowych dla granitu i dla bazaltu, stan popołudniowy (pomniejszony o liczbę wykorzystanej kostki w czasie pracy), dane dotyczące realizowanych dostaw.

Poranny stan zgromadzonej na rynku kostki w dniu 1.03.2013 roku wynosił odpowiednio dla granitu 500 m² i dla bazaltu 200 m². Liczba metrów układanej kostki zależy od porannego stanu kostki granitowej: wszyscy pracownicy (20 osób) układają kostkę granitową do dnia,

w którym rano zapas kostki granitowej jest mniejszy niż ten zużywany codziennie przez cały zespół 20 pracowników (90 m^2). Wówczas połowa pracowników zostaje oddelegowana do pracy z kostką bazaltową ($10 \cdot 3,8 \text{ m}^2 = 38 \text{ m}^2$), a połowa układa nadal kostkę granitową ($10 \cdot 4,5 \text{ m}^2 = 45 \text{ m}^2$). Kiedy zapas kostki granitowej zostanie uzupełniony, wszyscy ponownie układają tylko kostkę granitową. Powyższy opis odpowiada formule w arkuszu kalkulacyjnym:

$=\text{JEŻELI}(C6 \geq \$A\$1 * \$F\$2; \$A\$1 * \$F\$2; \$A\$1 / 2 * \$F\$2),$

która jest również przedstawiona na poniższym rysunku.

E6		=JEŻELI(C6>=\$A\$1*\$F\$2;\$A\$1*\$F\$2;\$A\$1/2*\$F\$2)								
	A	B	C	D	E	F	G	H	I	J
1	20	pracowników				granit	bazalt			
2	16500	suma metrów kostki			człowiek	4,5	3,8			
3					samochód	32	28			
4										
5	data	dni_tyg	rano grani	rano baz	ukł granit	ukł bazalt	po pr granit	po pr baz	dost granit	dost baz
6	2013-03-01	5	500	200	90	0	410	200	0	0
7	2013-03-04	1	410	200	90	0	320	200	0	28
8	2013-03-05	2	320	228	90	0	230	228	0	0
9	2013-03-06	3	230	228	90	0	140	228	0	28
10	2013-03-07	4	140	256	90	0	50	256	32	0

Po pracy brukarzy sytuacja stanu kostki się zmieniła. Od stanu porannego należy odjąć liczbę ułożonych metrów kwadratowych. Będzie to dla granitu formuła: $=C6-E6$, a dla bazaltu formuła: $=D6-F6$.

Przeanalizujemy dostawy kostki. Uzupełnianie zapasów następuje wieczorem, po pracy brukarzy. Jeżeli po pracy zapas kostki granitowej jest mniejszy niż 40 m^2 to przyjeżdżają 3 samochody z dostawą, jeżeli zapas wynosi od 40 m^2 do 100 m^2 włącznie – przyjeżdża 1 samochód, jeżeli zapas kostki jest większy niż 100 m^2 , wówczas nie ma dostawy. Dla obliczenia wielkości dostawy zastosujemy funkcję jeżeli:

$=\text{JEŻELI}(G6 < 40; 3 * \$F\$3; \text{JEŻELI}(G6 \leq 100; 1 * \$F\$3; 0))$

Zapasy kostki bazaltowej jest uzupełniany w każdy poniedziałek i środę z użyciem jednego samochodu, co możemy zapisać za pomocą funkcji jeżeli i zagnieżdżonej funkcji LUB:

$=\text{JEŻELI}(\text{LUB}(B6=3; B6=1); 28; 0)$

Jak będzie wyglądać stan kostki w następnym dniu roboczym rano? Pozostaje nam uwzględnić wieczorne dostawy. Dla granitu będzie to formuła: $=G6+I6$, a dla bazaltu: $=H6+J6$.

Wykonaliśmy w ten sposób całą symulację dobowych działań firmy Bruker. Teraz wystarczy wpisać formuły i funkcje skopiować aż do dnia 30.11.2013 roku.

Aby odpowiedzieć na polecenie a), należy sprawdzić liczbę dostaw bazaltu, czyli liczbę poniedziałków i śród w okresie od 1.03.2013 roku do 30.11.2013 roku, stosując funkcję LICZ.JEŻELI.

Polecenie b) sprawdza czy prawidłowo zbudowaliśmy formułę dotyczącą pracy z granitem. Do dnia 7.03.2013 stan kostki jest powyżej 100 m^2 i dopiero po pracy w tym dniu spada do

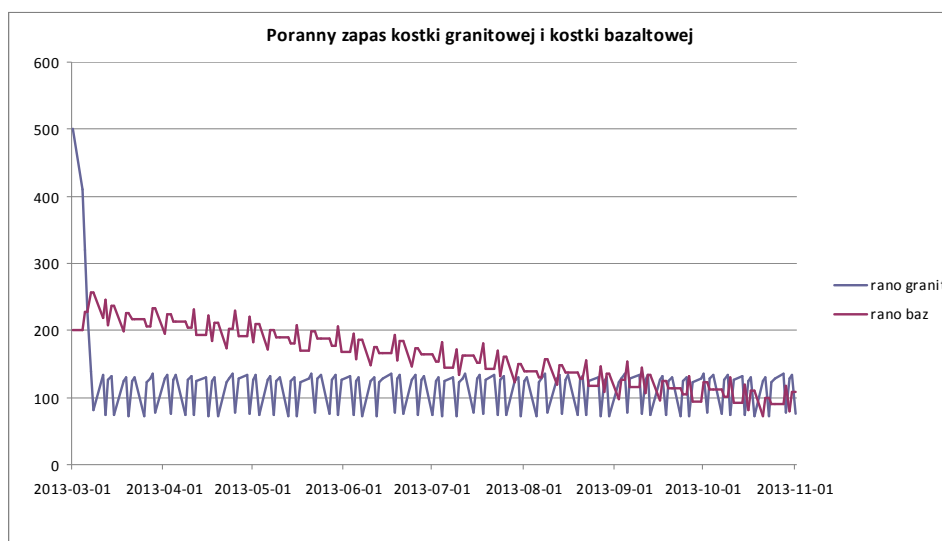
50 m², co skutkuje pierwszą dostawą kostki granitowej (1 samochód).

Dopiero polecenie c) kontroluje prawidłowe wykonanie całej symulacji. Aby obliczyć liczby metrów kwadratowych kostki granitowej oraz liczby metrów kwadratowych kostki bazaltowej ułożonych do dnia wskazanego datą włącznie (stan wieczorny), należy utworzyć kolumnę suma dla granitu i suma dla bazaltu (są to odpowiednio kolumny K oraz L na poniższym rysunku), w których będziemy dodawać kolejne metry kwadratowe ułożonej kostki.

K7 Σ =SUMA(E\$6:E7)

	E	F	G	H	I	J	K	L	M
1		granit	bazalt						
2	człowiek	4,5	3,8						
3	samochód	32	28						
4									
5	ukł granit	ukł bazalt	po pr granit	po pr baz	dost granit	dost baz	suma granit	suma baz	całosc kostki
6	90	0	410	200	0	0	90	0	90
7	90	0	320	200	0	28	180	0	180
8	90	0	230	228	0	0	270	0	270
9	90	0	140	228	0	28	360	0	360
10	90	0	50	256	32	0	450	0	450
11	45	38	37	218	96	0	495	38	533

W poleceniu d) należy utworzyć wykres liniowy porannych stanów kostki granitowej i bazaltowej. Należy pamiętać o prawidłowym zaznaczeniu zakresu danych i opisanie wykresu w sposób umożliwiający odczytanie stanu konkretnego rodzaju kostki w wybranym dniu.



Jeżeli podsumujemy liczbę ułożonej kostki granitowej oraz ułożonej kostki bazaltowej, otrzymamy liczbę metrów kwadratowych, które firma ułożyła do tego właśnie dnia włącznie. Okazuje się, że w dniu 19.11.2013 zostanie pokryta cała powierzchnia rynku starego miasta i firma Bruker zakończy pracę.

Na końcu musimy wspomnieć, że to zadanie niemal w całości można również rozwiązać, pisząc odpowiedni program. Jedyną problematyczną rzeczą jest utworzenie wykresu, który można wykonać w arkuszu kalkulacyjnym, zapisując wcześniej potrzebne dane do plików tekstowych.

Zadanie 18. Telefony (0–11)

Firma „Ciasteczko” wprowadziła na rynek nowy baton czekoladowy. Z tej okazji przeprowadziła konkurs SMS-owy. Zadanie konkursowe polega na przesłaniu odpowiedzi na pytanie: „**Czy smakuje Ci nasz nowy baton czekoladowy? Odpowiedz: Tak lub Nie.**”

Wiele osób, licząc na zwiększenie szansy wygranej, wysyłało SMS-y wielokrotnie. W pliku tekstowym o nazwie `telefony.txt` znajduje się 2000 zarejestrowanych numerów telefonów wraz z wysłaną odpowiedzią. Każdy numer telefonu i udzielona odpowiedź umieszczona jest w jednym wierszu, informacje rozdzielone są pojedynczym znakiem odstępu.

Korzystając z danych umieszczonych w pliku `telefony.txt`, wykonaj następujące polecenia. Odpowiedzi do poszczególnych podpunktów zapisz w pliku `wyniki_konkursu.txt` (poza wykresem do podpunktu a), a każdą z nich poprzedź literą oznaczającą ten podpunkt.

- a) Podaj, ile razy wysłano odpowiedź „Tak”, a ile razy odpowiedź „Nie”. Sporządź wykres procentowy ilustrujący otrzymane wyniki. Pamiętaj o prawidłowym i czytelnym opisie wykresu.
- b) Numery telefonów należą do czterech grup numeracyjnych rozpoczynających się cyframi: 5, 6, 7, 8. Ile numerów telefonów należy do każdej z grup? W swoim zestawieniu uwzględnij powtarzające się numery telefonów.
- c) Nagrodę I stopnia otrzymała osoba, w której numerze telefonu suma cyfr jest największa. Podaj numer telefonu oraz sumę cyfr numeru.
- d) Najdłuższym malejącym numerem telefonu nazywamy taki numer, którego początkowe cyfry tworzą najdłuższy malejący ciąg, tzn. kolejna jego cyfra, począwszy od drugiej, jest mniejsza od cyfry ją poprzedzającej, np. w numerze 654209192 pięć pierwszych cyfr tworzy malejący ciąg, zaś w numerze 865320542 sześć pierwszych cyfr tworzy malejący ciąg. Do nagrody II stopnia wybrano te numery telefonów, których cyfry tworzą najdłuższy malejący ciąg. Podaj numery telefonów, które otrzymały tą nagrodę.
- e) Firma postanowiła wręczyć nagrodę pocieszenia właścicielowi tego numeru, z którego wysłano najwięcej SMS-ów. Jaką największą liczbę SMS-ów wysłano z jednego numeru?

Do oceny oddajesz plik(i) o nazwie ,
tu wpisz nazwę(y) pliku(ów)
 zawierający(e) komputerową(e) realizację(e) Twoich obliczeń, plik tekstowy
`wyniki_konkursu.txt` z odpowiedziami do punktów a–e (odpowiedź do każdego
 podpunktu powinna być poprzedzona jego nazwą) oraz plik ,
tu wpisz nazwę pliku
 zawierający wykres do podpunktu a.

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>4. Opracowywanie informacji za pomocą komputera, w tym: rysunków, tekstów, danych liczbowych, animacji, prezentacji multimedialnych i filmów.</i> <i>Zdający:</i> <i>4) wykorzystuje arkusz kalkulacyjny do obrazowania zależności funkcyjnych i do zapisywania algorytmów.</i> <i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin;</i> <i>2) stosuje podejście algorytmiczne do rozwiązywania problemu;</i> <i>21) przeprowadza komputerową realizację algorytmu i rozwiązania problemu.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za podanie poprawnej liczby odpowiedzi "Tak" oraz poprawnej liczby odpowiedzi "Nie" – 1 punkt. Za poprawny wykres – 2 punkty, w tym: - poprawny dobór danych i typ wykresu – 1 punkt, - czytelny opis wykresu – 1 punkt.	3	11
b	Za podanie poprawnej liczby numerów telefonów dla czterech grup – 2 punkty. Za podanie poprawnej liczby numerów telefonów dla co najmniej dwóch grup – 1 punkt.	2	
c	Za podanie poprawnego numeru telefonu – 1 punkt. Za podanie poprawnej sumy cyfr dla poprawnego numeru – 1 punkt.	2	
d	Za podanie poprawnych trzech numerów – 3 punkty, po 1 punkcie za każdy poprawny numer.	3	
e	Za podanie poprawnej liczby – 1 punkt.	1	

Zadanie 18. Telefony (0–11) – rozwiązanie

Pliki zawierające rozwiązania znajdują się w folderze TELEFONY.

Komentarz

Większość podpunktów zadania sprowadza się do zliczania wierszy spełniających określone kryteria. Naturalnym narzędziem do rozwiązania takiego zadania jest język programowania bądź arkusz kalkulacyjny. Ponieważ podpunkty b, c oraz d wymagają dostępu nie tylko do całych pól naszej tabeli (czyli numer telefonu i odpowiedź), ale do poszczególnych cyfr numeru, wygodniejszym narzędziem wydaje się język programowania. Niemniej rozwiązanie zadania jest również możliwe w oparciu o arkusz kalkulacyjny. Poniżej omówimy dwa różne rozwiązania: w pierwszym napiszemy program w języku C, a w drugim korzystać będziemy z funkcji i innych narzędzi dostępnych w MS Excel.

Rozwiązanie 1: język programowania.

Korzystając z dostępnych operacji wejścia/wyjścia z wykorzystaniem plików (lub strumieni), możemy zapisać tabelę z pliku w tablicach. W naszym rozwiązaniu numery telefonów zapisujemy jako napisy w tablicy `num`, natomiast odpowiadające im odpowiedzi („Tak”/„Nie”) reprezentujemy w tablicy liczb całkowitych `odp`, gdzie 1 odpowiada wartości „Tak” a 0 wartości „Nie”:

```
#define n 2000 // liczba wierszy
typedef char napis9[10]; // typ dla napisow o długości 9

int odp[n]; // odp[i]=1/0 gdy odpowiedz „Tak”/„Nie” w i-tym wierszu
napis9 num[n]; // num[i] to numer telefonu w i-tym wierszu
```

Podpunkt a

Przy powyższej reprezentacji danych punkt a) sprowadza się do policzenia sumy `odp[0]+odp[1]+...+odp[1999]`.

Podpunkt b

Aby wyznaczyć liczbę numerów zaczynających się od cyfry i dla $i=5, 6, 7, 8$, utworzyliśmy tablicę liczb całkowitych `grupa`, gdzie `grupa[i]` ma spełniać rolę licznika dla numerów zaczynających się od i . Przeglądając w pętli wartości `num[j][0]` dla $j=0, 1, \dots, 1999$, uzupełniliśmy odpowiednio wartości liczników. Aby zamienić znak reprezentowany przez `num[j][0]` na odpowiadającą mu cyfrę, użyliśmy podstawienia:

```
cyf=num[j][0] - '0';
```

Podpunkt c

Utworzyliśmy osobną funkcję `sumaCyfr`, która w pętli zlicza sumę cyfr liczby podanej jako ciąg cyfr w tablicy typu `napis9`. Z użyciem tej funkcji punkt c) sprowadzony został do klasycznego problemu wyznaczania największej wartości w ciągu.

Podpunkt d

W podpunkcie d również zastosowaliśmy modularyzację, wydzielając osobno funkcję `malejacy`, której wynikiem jest liczba elementów najdłuższego ciągu malejącego,

złożonego z początkowych cyfr ciągu podanego na wejściu. Używając tej funkcji:

- najpierw wyznaczyliśmy `maxD`, największą długość malejącego ciągu złożonego z początkowych cyfr numeru telefonu,
- następnie przejrzelśmy raz jeszcze wszystkie numery telefonów, wypisując jako wynik te numery, których początkowe `maxD` cyfr tworzy ciąg malejący.

Podpunkt e

Polecenie w podpunkcie e) jest o tyle trudne, że zliczanie wystąpień poszczególnych numerów wymaga wielokrotnego przeglądania listy telefonów. W naszym rozwiązaniu zliczamy wystąpienia kolejnych numerów, stosując dwie zagnieżdżone pętle. Zastosowaliśmy jednak drobną optymalizację. Wspomogliśmy się dodatkową tablicą `liczony`, ustawiając `liczony[i]` na 1 tylko wtedy, gdy *i*-ty numer jest równy numerowi występującemu wcześniej. Mogliśmy wówczas pominąć zliczanie tych elementów, dla których wartość w tablicy `liczony` była równa 1.

Można sobie oczywiście wyobrazić inne rozwiązania tego podpunktu. Gdyby numery telefonów były uporządkowane w pliku wejściowym, mielibyśmy pewność, że wszystkie wystąpienia danego numeru pojawiają się „obok siebie” (w kolejnych wierszach). A to z kolei ułatwiłoby znacznie zliczanie wystąpień poszczególnych numerów. Można zatem zacząć od sortowania ciągu numerów lub np. zastosować funkcje sortujące dostępne z biblioteki `algorithm`, dostępnej w większości kompilatorów języka C. W naszym rozwiązaniu przyjęliśmy jednak, że samodzielna implementacja sortowania byłaby utrudnieniem sobie zadania, natomiast wykorzystanie bibliotecznych funkcji sortujących pozostawiamy jako ćwiczenie dla czytelnika (wygodniej przy tym byłoby reprezentować numery jako tablicę liczb, nie napisów).

Rozwiązanie 2: arkusz kalkulacyjny.

Omówimy teraz rozwiązanie utworzone w oparciu o MS Excel. Po zaimportowaniu danych z pliku tekstowego przyjmujemy, że numery telefonów i odpowiedzi „Tak”/„Nie” znajdują się odpowiednio w kolumnach A i B arkusza, w wierszach 2, 3, ..., 2001 (dodajemy nowy wiersz na początku dla opisu kolumn). Rozwiązania punktów a), b) i e) możemy wówczas uzyskać z pomocą funkcji `LICZ.JEŻELI`.

Podpunkt a

Aby zliczyć liczbę odpowiedzi „Tak”, dla funkcji `LICZ.JEŻELI` wskazujemy obszar `B2:B2001` oraz jako kryterium wyszukiwania podajemy „Tak”:
`=LICZ.JEŻELI(B2:B2001;"Tak")`

Podpunkty b – d

Rozwiązanie dla podpunktów b), c) i d) zadania wymaga wydzielenia cyfr kolejnych numerów telefonów. Można do tego wykorzystać bardziej ogólną funkcję `FRAGMENT.TEKSTU`, która pozwala „wyciąć” z podanego tekstu fragment o wskazanej długości i od podanej pozycji. Na przykład wartością formuły
`=FRAGMENT.TEKSTU(A2;2;1)` będzie fragment tekstu z komórki A2, zaczynający się od drugiej pozycji, o długości jeden (czyli drugi znak z tekstu w A2). Co więcej, jeśli w komórce A2 znajduje się liczba, funkcja „automatycznie” przekształci ją na tekst przed wydzieleniem odpowiedniego fragmentu. W naszym rozwiązaniu „zautomatyzowaliśmy” proces wydzielania wszystkich 9 cyfr z kolejnych numerów w następujący sposób:

- w komórkach D1...L1 umieszczone zostały cyfry 1, 2, ..., 9,
- w komórce D2 wpisana została formuła: = FRAGMENT.TEKSTU(\$A2;D\$1;1),
- skopiowanie tej formuły do całego bloku D2:L1999 pozwoliło wydzielić wszystkie cyfry z każdego numeru.

Zwróćmy uwagę na sposób adresowania w formule z komórki D2:

- adres \$A2 gwarantuje, że po skopiowaniu formuły nie zmieni się kolumna, z której pobieramy tekst,
- adres D\$1 z kolei gwarantuje, że numer pozycji, od której zaczyna się wycinany fragment tekstu, zawsze pobierany będzie z pierwszego wiersza.

W rezultacie kolejne cyfry liczb z kolumny A umieścimy w kolumnach D...L. Nie możemy jednak wykonywać na nich operacji arytmetycznych, ponieważ wynikiem funkcji FRAGMENT.TEKSTU jest napis, nie liczba. Zastosujemy zatem funkcję WARTOŚĆ, która „zamienia” tekst na liczbę. Wzorcowa formuła w komórce D2 będzie więc wyglądać następująco:

= WARTOŚĆ (FRAGMENT.TEKSTU(\$A2;D\$1;1))

Podpunkt b

Aby zliczyć liczbę numerów telefonów zaczynających się od wskazanej cyfry, wykorzystamy fakt, że w kolumnie D wydzieliśmy pierwsze cyfry numerów telefonów. Aby uzyskać liczbę numerów zaczynających się od 5, możemy użyć formuły: =LICZ.JEŻELI(A2:A2001;5). Aby uniknąć kilkukrotnego „ręcznego” wpisywania formuły wg powyższego schematu, możemy zastosować adresowanie pośrednie i wskazać jako kryterium komórki, w których kolejno umieścimy liczby 5, 6, 7 i 8. Poniżej zamieszczamy przykładowe rozwiązanie zapisane w komórkach A9..B12 drugiego arkusza.

	A	B
5		
6		
7		
8	b)	
9	5	=LICZ.JEŻELI(arkusz1!\$D\$2:\$D\$2001;A9)
10	6	=LICZ.JEŻELI(arkusz1!\$D\$2:\$D\$2001;A10)
11	7	=LICZ.JEŻELI(arkusz1!\$D\$2:\$D\$2001;A11)
12	8	=LICZ.JEŻELI(arkusz1!\$D\$2:\$D\$2001;A12)
13		

Formuły w komórkach odpowiadających wartościom 6, 7, 8 i 9 możemy otrzymać, kopiując formułę wpisaną dla cyfry 5.

Podpunkt c

Aby rozwiązać podpunkt c, wystarczy zsumować wartości z kolumn D...L w każdym wierszu, wyznaczyć największą z tych sum i wyszukać numer odpowiadający tej sumie.

Podpunkt d

W podpunkcie d) zapiszemy przy pomocy formuł następujący algorytm wyznaczania długości najdłuższego ciągu malejącego, będącego początkiem wskazanego ciągu $c_1, c_2, \dots, c_k$:

Krok 1: Jeśli $c_1 > c_2$ to $r_2 \leftarrow 1$

w przeciwnym przypadku $r_2 \leftarrow 0$

Krok 2: Powtarzaj dla $i=3, 4, \dots, k$:

Jeśli $(c_{i-1} > c_i)$ oraz $(r_{i-1} = 1)$ to $r_i \leftarrow 1$

w przeciwnym przypadku $r_i \leftarrow 0$

Krok 3: Zwróć $(1 + r_2 + r_3 + \dots + r_k)$

W naszym przypadku wartość k z powyższego algorytmu to 9, a pierwszy ciąg znajduje się w komórkach D2, E2, ..., L2. Formuły wyznaczające wartości $r_2, \dots, r_9$ w komórkach N2, ..., U2 realizujące powyższy algorytm mogą wyglądać następująco:

– unikalna formuła w komórce N2:

=JEŻELI (D2>E2;1;0)

– formuła w komórce O2, którą kopiujemy do P2, ..., U2:

=JEŻELI (N2=1;JEŻELI (E2>F2;1;0);0)

Następnie możemy:

- skopiować powyższe formuły do kolejnych wierszy arkusza,
- w kolumnie V każdego wiersza wyliczyć wartość jeden plus największa z wartości $r_2, r_3, \dots, r_9$ (czyli maksimum wartości z kolumn N, M, ..., U tego wiersza),
- wybrać największą wartość z kolumny V,
- wyszukać numery telefonów z wierszy, w których pojawia się ta największa wartość w kolumnie V.

Podpunkt e

Po raz kolejny zastosujemy funkcję LICZ.JEŻELI, umiejętnie wykorzystując też adresowanie pośrednie. Jeśli w komórce C2 arkusza z danymi wpisemy formułę =LICZ.JEŻELI (\$A\$2:\$A\$2001;A2), wówczas uzyskamy nie tylko liczbę wystąpień pierwszego numeru (znajdującego się w komórce A2) w bloku A2:A2001, ale po skopiowaniu tej formuły do kolejnych komórek kolumny C wyznaczymy liczby wystąpień kolejnych numerów.

Podpunkty a) i e) zadania nie wymagają wydzielania elementów (cyfr) numerów telefonów, można je więc zaliczyć do klasycznych ćwiczeń z zakresu analizowania i filtrowania danych, które bez większych trudności można zrealizować w wybranym arkuszu kalkulacyjnym.

W pozostałych punktach zadania uzyskanie wyniku zasadniczo wymaga dostępu do poszczególnych cyfr numerów telefonów. Co prawda wyniki punktu b) zależą jedynie od pierwszych cyfr a zatem możliwe jest rozwiązanie w oparciu o porównania całych numerów (np. numery zaczynające się od 5 są nie mniejsze niż 500000000 i nie większe niż 599999999), jednak w punktach c) i d) wyniki zależą od wszystkich cyfr w numerach telefonów. Dlatego rozwiązanie z wykorzystaniem języka programowania wydaje się bardziej naturalne. Zaprezentowaliśmy jednak również rozwiązanie w arkuszu kalkulacyjnym, aby zademonstrować, że stosując odpowiednie formuły można „implementować” proste algorytmy, gdzie komórki arkusza służą do przechowywania wartości zmiennych w kolejnych etapach obliczeń. Niemniej szczególnie w podpunkcie d) (gdzie konieczne było ustalanie najdłuższego malejącego ciągu początkowych cyfr numeru), nasze rozwiązanie wymagało niestandardowych i dość nienaturalnych pomysłów.

W zaprezentowanym rozwiązaniu programistycznym numery telefonów reprezentowane są jako ciągi znaków, co ułatwia dostęp do poszczególnych cyfr. Warto wspomnieć, że reprezentacja numerów w postaci liczb również daje wygodny dostęp do cyfr. Zauważmy, że dla zmiennej `n` typu `int` wartość `n%10` jest równa ostatniej jej cyfrze, natomiast `n/10` to wartość uzyskana po usunięciu z zapisu dziesiętnego `n` ostatniej cyfry (pamiętajmy, że w języku C wartość `n/10` dla zmiennej `n` typu `int` jest równa wynikowi dzielenia całkowitego `n` przez 10). Taki sposób dostępu do cyfr numeru jest jednak niewygodny w podpunkcie d) zadania, gdzie konieczne jest „przeglądanie” kolejnych cyfr w kolejności od pierwszej (najbardziej znaczącej), a nie od ostatniej (najmniej znaczącej).

Na końcu należy podkreślić, że zadanie nie narzuca żadnego sposobu rozwiązania, więc nic nie stoi na przeszkodzie, aby odpowiednio dobrać takie narzędzia informatyczne do rozwiązania zadań w kolejnych podpunktach, aby część z nich rozwiązać przy pomocy języka programowania, a część przy pomocy arkusza kalkulacyjnego.

Zadanie 19. Korek (0–11)

W plikach `sprzedaz.txt`, `produkt.txt` i `kategoria.txt` znajdują się informacje o sprzedaży produktów z korka w sklepie *Koreczek*. Pierwszy wiersz każdego z plików jest wierszem nagłówkowym, a dane w wierszach rozdzielone są znakami tabulacji.

W pliku `sprzedaz.txt` znajduje się 2200 wierszy z informacjami o sprzedanych produktach w ciągu całego roku 2012: *Id_zakupu*, *Id_produktu*, *Data_zakupu*, *Ilosc* (liczba sprzedanych jednostek danego produktu).

Przykład:

<i>Id_zakupu</i>	<i>Id_produktu</i>	<i>Data_zakupu</i>	<i>Ilosc</i>
1	p34	2012-08-09	9
2	p87	2012-08-07	12
3	p86	2012-08-03	26
4	p20	2012-01-05	2

W pliku `produkt.txt` znajduje się 99 wierszy z informacjami o produktach: *Id_produktu*, *Nazwa*, *Cena*, *Jednostka*, *Id_kategoria*.

Przykład:

<i>Id_produktu</i>	<i>Nazwa</i>	<i>Cena</i>	<i>Jednostka</i>	<i>Id_kategoria</i>
p1	Especial_Big	24,99	m2	k1
p2	Toledo_Natural	23,99	m2	k1
p3	Toledo_Red	23,99	m2	k1

W pliku `kategoria.txt` znajduje się 21 wierszy z opisem kategorii sprzedawanych produktów: *Id_kategoria*, *Nazwa* (nazwa kategorii do której należy grupa produktów).

Przykład:

<i>Id_kategoria</i>	<i>Nazwa</i>
k1	korek_scienny
k2	podklad_korkowy
k3	granulat_korkowy

Wykorzystując dane zawarte w tych plikach i dostępne narzędzia informatyczne, wykonaj poniższe polecenia. Odpowiedzi do poszczególnych podpunktów zapisz w pliku tekstowym o nazwie `wyniki_korek.txt`. Odpowiedź do każdego podpunktu poprzedź literą oznaczającą ten podpunkt.

- Podaj zestawienie zawierające informacje o liczbie dokonanych zakupów w każdym miesiącu. Zestawienie posortuj rosnąco ze względu na liczbę zakupów.
- Dla każdej kategorii oblicz łączną wartość sprzedanych produktów. Utwórz zestawienie o kolumnach: nazwa kategorii, łączna wartość sprzedanych produktów w danej kategorii. Zestawienie uporządkuj alfabetycznie według nazw.
- Podaj nazwę produktu z kategorii *wyroby_korkowe*, którego sprzedano najwięcej oraz wielkość jego sprzedaży.
- Podaj zestawienie zawierające dla każdego produktu z kategorii *parkiet_korkowy* i *panele_korkowe* informacje o łącznej liczbie m² sprzedanego produktu.

Do oceny oddajesz plik(i) o nazwie, zawierający
 tu wpisz nazwę pliku(ów)
 komputerową realizację Twoich obliczeń oraz plik tekstowy wyniki_korek.txt,
 zawierający odpowiedzi do wszystkich podpunktów zadania. Odpowiedź do każdego
 podpunktu w pliku wyniki_korek.txt powinna być poprzedzona nazwą podpunktu.

Wymagania ogólne	<i>II. Wyszukiwanie, gromadzenie i przetwarzanie informacji z różnych źródeł; opracowywanie za pomocą komputera: rysunków, tekstów danych liczbowych, motywów, animacji, prezentacji multimedialnych.</i>
Wymagania szczegółowe	<i>2. Wyszukiwanie, gromadzenie, selekcjonowanie, przetwarzanie i wykorzystywanie informacji, korzystanie z różnych źródeł i sposobów zdobywania informacji.</i> <i>Zdający:</i> <i>1) projektuje relacyjną bazę danych z zapewnieniem integralności danych,</i> <i>2) stosuje metody wyszukiwania i przetwarzania informacji w relacyjnej bazie danych,</i> <i>3) tworzy aplikację bazodanową, wykorzystującą język zapytań, kwerendy, raporty; zapewnia integralność danych na poziomie pól, tabel, relacji.</i>

Schemat punktowania

Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za podanie poprawnego zestawienia zawierającego informacje o liczbie zakupów w każdym miesiącu – 1 punkt. Za posortowanie otrzymanego zestawienia ze względu na liczbę zakupów – 1 punkt.	2	10
b	Za podanie poprawnego zestawienia zawierającego nazwy kategorii i łączną wartość sprzedanego towaru dla każdej kategorii – 2 punkty. Za podanie zestawienia zawierającego poprawne wartości dla co najmniej 10 kategorii – 1 punkt. Za posortowanie otrzymanego zestawienia alfabetycznie – 1 punkt.	3	
c	Za podanie poprawnej nazwy produktu – 1 punkt. Za podanie poprawnej liczby sprzedanego produktu – 1 punkt.	2	
d	Za poprawne zestawienie zawierające łączną liczbę sprzedanych produktów w każdej z podanych kategorii – 3 punkty. Za zestawienie zawierające błędny jeden wiersz albo jedną kolumnę – 1 punkt.	3	

Zadanie 19. Korek (0–11) – rozwiązanie

Pliki zawierające rozwiązania znajdują się w folderze KOREK.

Komentarz

Cechą wyróżniającą zadanie „Korek” spośród pozostałych zadań jest konieczność powiązania ze sobą danych rozmieszczonych w trzech różnych plikach. Każdy z tych plików może odpowiadać innej tabeli relacyjnej bazy danych, w której mamy następujące powiązania typu „jeden do wielu”:

- table `produkt.txt` i `sprzedaz.txt` wiąże pole `ID_produkt` w `produkt.txt` i pole `Id_produktu` w `sprzedaz.txt`;
- table `kategoria.txt` i `produkt.txt` wiąże pole `Id_kategoria` z obu tabel.

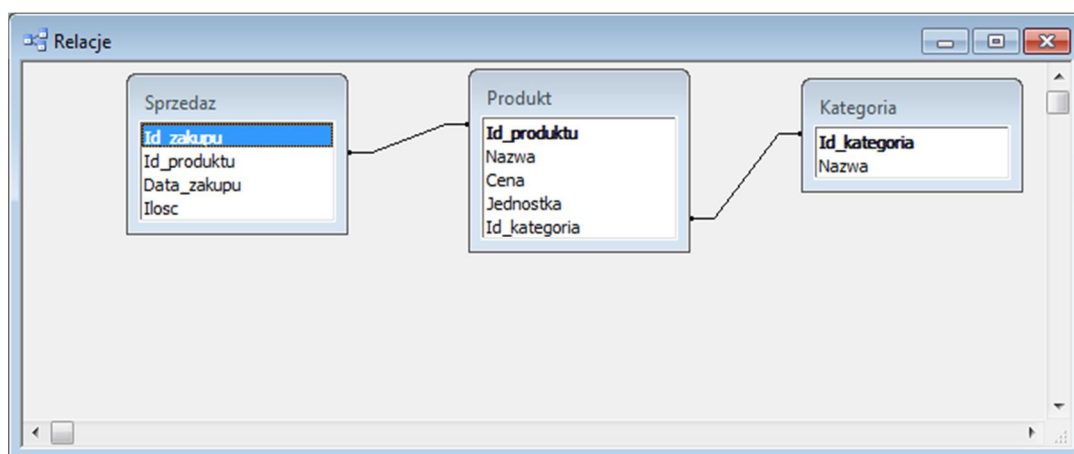
Naturalnym narzędziem do analizy takich danych jest aplikacja służąca do zarządzania bazą danych, np. MS Access. Poniżej zaprezentujemy rozwiązanie z użyciem tego narzędzia.

Aplikacja bazodanowa wymaga dość dużego „wstępnego” nakładu pracy związanego z zakładaniem tabel, ustalaniem typów i rozmiarów pól, powiązań (relacji między tabelami). Dlatego warto czasem spróbować alternatywnych rozwiązań, na przykład w arkuszu kalkulacyjnym. Zaprezentujemy taką próbę na przykładzie zadania „Korek”, pozostawiając czytelnikom ocenę, które narzędzie jest lepiej dopasowane do specyfiki zadania.

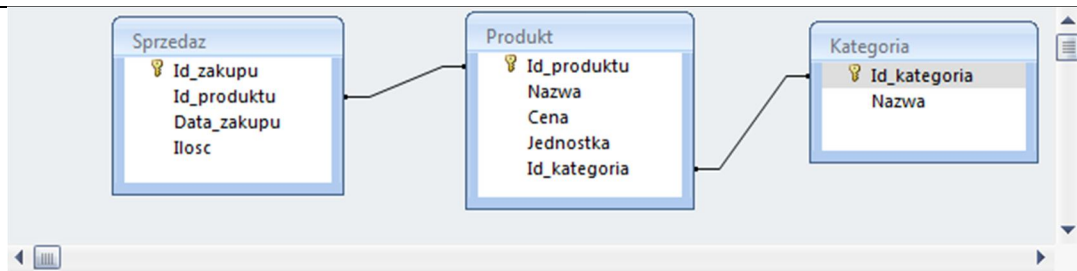
Rozwiązanie 1: baza danych (MS Access)

Przed przystąpieniem do rozwiązania podpunktów a) – d) zadania, musimy założyć table odpowiadające trzem plikom z danymi. Wygodnym rozwiązaniem jest utworzenie ich poprzez import danych (Plik → Pobierz dane zewnętrzne → Importuj w MS Office 2003 lub Dane Zewnętrzne → Importowanie → Plik tekstowy w MS Office 2007) i skorzystanie z kreatora importu tekstu. Przyjmijmy, że po zakończeniu tego procesu mamy table `Sprzedaz`, `Produkt` i `Kategoria`, a pola tych tabel mają takie nazwy, jak podano w przykładach w treści zadania.

Następnie ustalimy relacje między tabelami i w efekcie uzyskujemy następujący schemat bazy danych⁴:



⁴ Rysunki w niniejszym opisie pochodzą z MS Office 2003 i MS Office 2007.



Teraz możemy przystąpić do rozwiązania podpunktów a) – d) zadania, tworząc odpowiednie kwerendy

Podpunkt a

Wszystkie dane do podpunktu a znajdują się w tabeli *Sprzedaz*, brakuje w niej jednak pola identyfikującego miesiąc każdej sprzedaży. Dodajemy więc pole wyliczane, w którym wyznaczamy miesiąc sprzedaży w oparciu o wartość pola *Data_zakupu*. Co prawda składnia wyrażeń definiujących pola wyliczane jest dość skomplikowana, jednak w MS Access możemy tworzyć takie pola za pomocą konstruktora wyrażeń, w którym wybieramy poszczególne elementy wyrażenia z odpowiednich list.

Aby wyznaczyć liczbę transakcji w każdym miesiącu, grupujemy dane według nowo utworzonego pola identyfikującego miesiąc sprzedaży, zliczamy liczbę wierszy w każdej grupie (funkcja *Policz*) oraz sortujemy wynikowe zestawienie wg tych wyliczonych wartości. Szczegóły prezentujemy na załączonym obrazie okna projektu kwerendy:

The screenshot shows the 'a : Kwerenda wybierająca' (a : Select Query) window in MS Access. The design grid is as follows:

Pole:	Wyrażenie	Tabela	Podsumowanie	Sortuj	Pokaż	Kryteria	lub
	Wyr 1: MonthName(Month([Sprzedaz]![Data_zakupu]);1)						
		Id_zakupu					
		Sprzedaz					
			Grupuj według				
			Policz				
				Rosnąco			
					☒		
						☒	
							☐

The screenshot shows the Microsoft Access interface. At the top, the 'Sprzedaz' table is displayed with fields: Id_zakupu (primary key), Id_produktu, Data_zakupu, and Ilosc. Below this, a query design grid is visible. The grid has columns for 'Pole' (Field), 'Tabela' (Table), 'Podsumowanie' (Summary), 'Sortuj' (Sort), 'Pokaż' (Show), 'Kryteria' (Criteria), and 'lub' (Or). The first row of the grid contains the following data:

Pole:	Tabela:	Podsumowanie:	Sortuj:	Pokaż:	Kryteria:	lub:
Wyr1: MonthName(Month([Sprzedaz]![Data_zakupu]);1)	Sprzedaz	Grupuj według		☒		

Podpunkt b

Zanim będziemy mogli sumować wartości sprzedaży w obrębie kategorii produktów, musimy wyznaczyć wartość każdej sprzedaży, co wymaga dostępu do tabel Sprzedaz i Produkt. W tym celu tworzymy pomocniczą kwerendę, w której korzystamy z powiązania obu tabel i tworzymy pole wyliczane, w którym wartość Ilosc z każdej sprzedaży mnożymy przez wartość Cena z tabeli Produkt:

The screenshot shows the Microsoft Access interface with a query named 'wartość_zakupów : Kwerenda wybierająca'. The design grid shows a relationship between the 'Produkt' and 'Sprzedaz' tables. The 'Produkt' table has fields: Id_produktu (primary key), Nazwa, Cena, Jednostka, and Id_kategoria. The 'Sprzedaz' table has fields: Id_zakupu (primary key), Id_produktu, Data_zakupu, and Ilosc. The design grid is as follows:

Pole:	Tabela:	Podsumowanie:	Sortuj:	Pokaż:	Kryteria:	lub:
Id_zakupu	Sprzedaz	Grupuj według		☒		
Wyr1: [Sprzedaz]![Ilosc]*[Produkt]![Cena]		Grupuj według		☒		

This screenshot is identical to the previous one, showing the 'wartość_zakupów : Kwerenda wybierająca' query design grid. It shows the relationship between the 'Produkt' and 'Sprzedaz' tables and the calculated field 'Wyr1: [Sprzedaz]![Ilosc]*[Produkt]![Cena]'.

Następnie wystarczy zsumować wartości poszczególnych sprzedaży w obrębie kategorii produktów, co możemy zrobić tworząc sprzężenie między tabelą *Kategoria* i nowo założoną kwerendą poprzez pole *Id_kategoria*. Wynikową kwerendę prezentujemy poniżej.

b : Kwerenda wybierająca

Pole:	Wyr1		
Tabela:	wartość_zakupów		
Podsumowanie:	Suma		
Sortuj:	Rosnąco		
Pokaż:	☒	☒	☐
Kryteria:			
lub:			

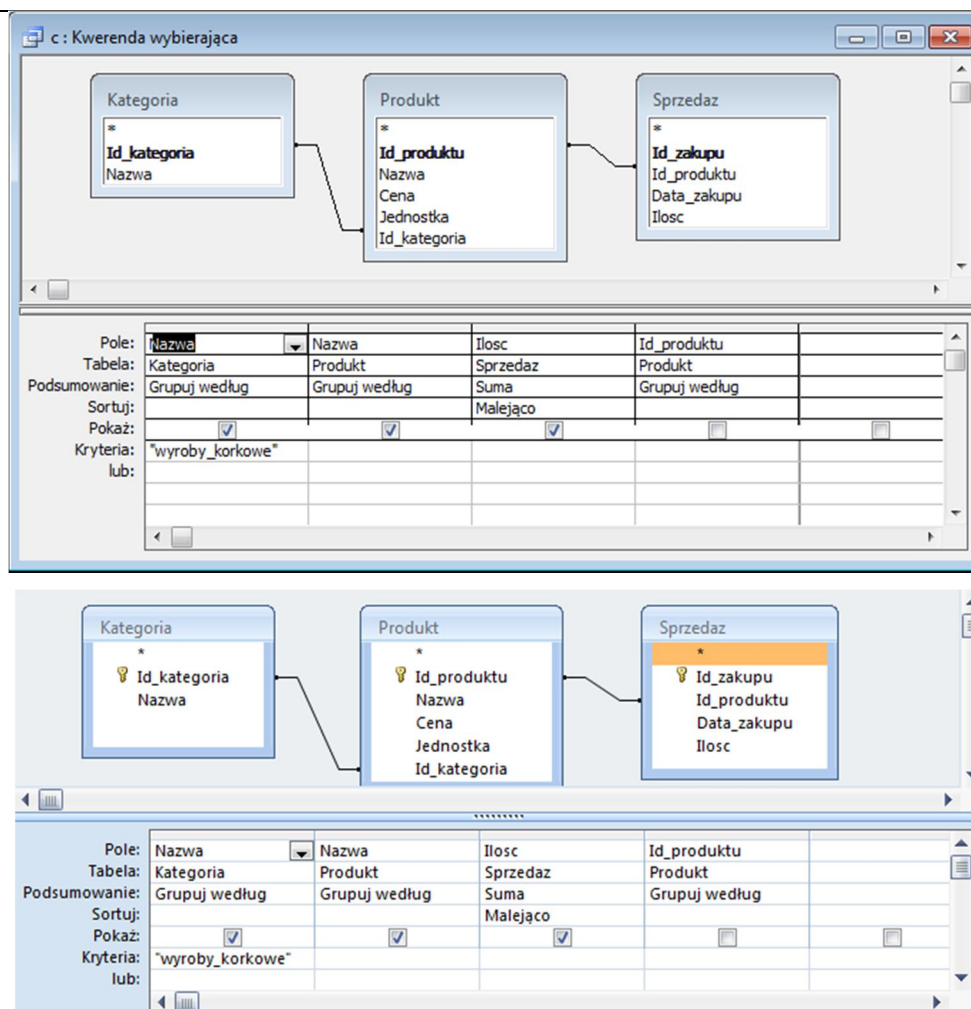
Kwerenda złączeniowa

Pole:	Nazwa	Wyr1	
Tabela:	Kategoria	wartość_zakupów	
Suma:	Grupuj według	Suma	
Sortuj:	Rosnąco		
Pokaż:	☒	☒	☐
Kryteria:			
lub:			

Podpunkt c

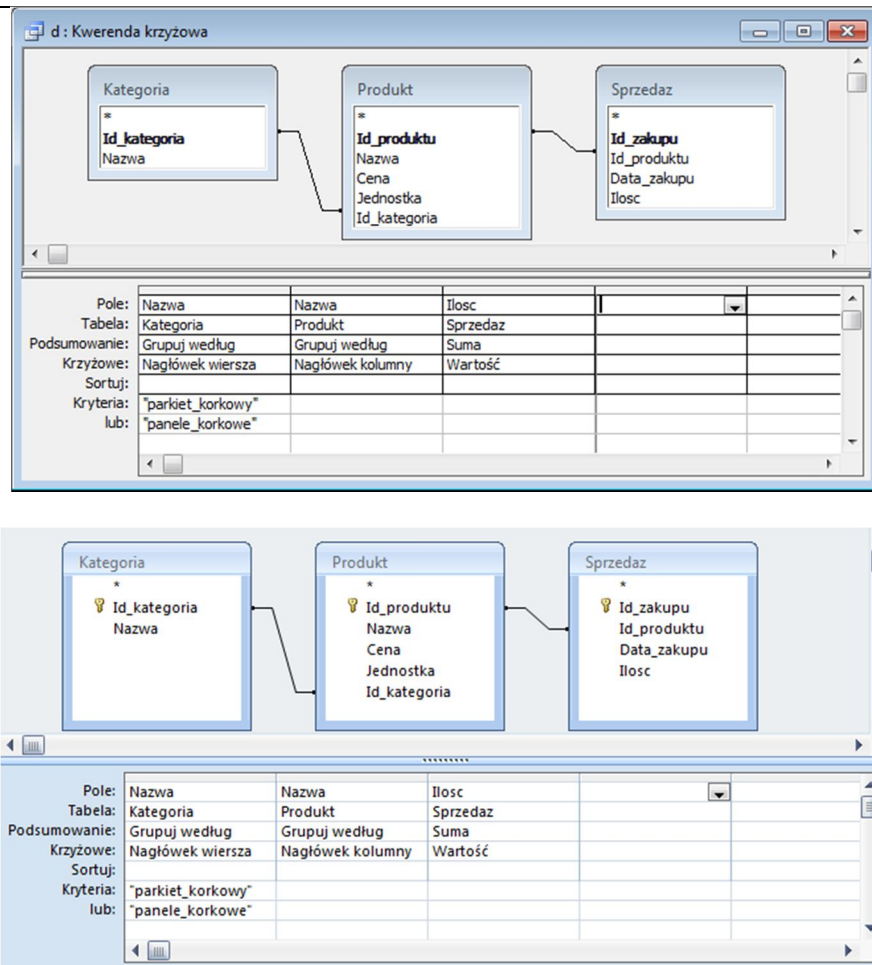
Aby ustalić ilość jednostek sprzedaży każdego produktu, wystarczyłoby pogrupować tabelę *Sprzedaz* wg pola *Id_produktu* i wyznaczyć sumę wartości pola *Ilosc* w każdej grupie. Moglibyśmy też posortować zestawienie wg wyliczonych sum.

Takie rozwiązanie nie daje jednak dostępu do nazw produktów, nie pozwala też wydzielić produktów z kategorii *wyroby_korkowe*. Dlatego do rozwiązania punktu c) wykorzystamy wszystkie trzy tabele wraz z łączącymi je relacjami. Aby w wyniku uzyskać tylko produkty z kategorii *wyroby_korkowe*, zastosujemy filtrowanie wpisując frazę „*wyroby_korkowe*” w wierszu kryteriów dla nazwy kategorii. Przykładowy projekt kwerendy załączamy poniżej.

**Podpunkt d**

Chcielibyśmy uzyskać zestawienie, w którego wierszach znajdują się nazwy kategorii, w kolumnach nazwy produktów, a na przecięciach wierszy i kolumn wielkości sprzedaży odpowiednich produktów. Zastosujemy w tym celu kwerendę krzyżową, której funkcjonalność jest podobna do tabel przestawnych w MS Excel. Poniżej prezentujemy nasze rozwiązanie dla punktu d), w którym chcielibyśmy zwrócić uwagę na następujące elementy:

- w wierszu o nazwie „Krzyżowe” wskazujemy rolę poszczególnych pól w kwerendzie krzyżowej: nagłówków kolumn, nagłówków wierszy, wartości uwzględnianych w obliczeniach,
- w wierszach „Kryteria” kolumny Kategoria wpisane zostały kategorie, do których ograniczone miały być wyniki punktu d) zadania.



Rozwiązanie 2: arkusz kalkulacyjny (MS Excel)

Rozwiązanie zadania w arkuszu kalkulacyjnym rozpoczniemy od wczytania zawartości plików `sprzedaz.txt`, `produkt.txt` i `kategoria.txt` do trzech różnych arkuszy o nazwach `sprzedaz`, `produkt` i `kategoria` z takimi nagłówkami kolumn, jak podane w treści zadania. Możemy to zrobić, importując odpowiednie pliki tekstowe w MS Excel (Dane → Importuj dane zewnętrzne → Importuj dane w MS Office 2003 lub Dane → Dane zewnętrzne → Z tekstu w MS Office 2007), każdy do innego arkusza.

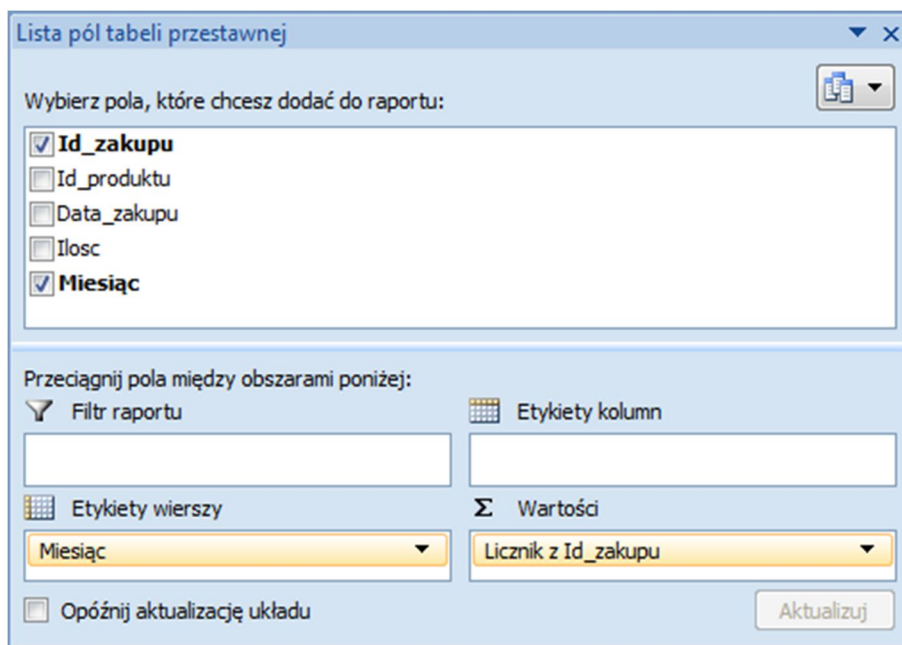
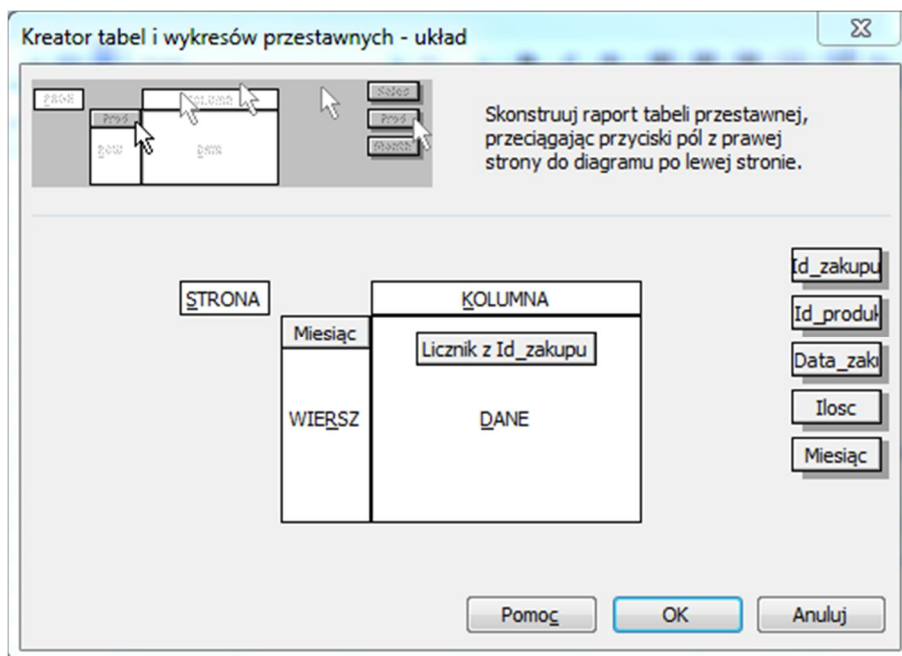
Rozwiązując kolejne podpunkty zadania, tworzyć będziemy tabele przestawne w oparciu o dane z zaimportowanych tabel. Aby utworzyć tabelę przestawną, wskazujemy:

1. kolumny względem których grupowane będą dane,
2. kolumnę, z której brane będą wartości do obliczeń,
3. funkcję, którą stosować będziemy na wartościach z kolumny podanej w 2.

Powyższe parametry ustalamy w projekcie tabeli przestawnej, ustalając wartości obszarów Etykiety wierszy, Etykiety kolumn i Wartości (terminy takie stosowane są w MS Office 2007; ich odpowiednikami w MS Office 2003 są obszary Wiersz, Kolumna i Dane). Obszary Etykiety wierszy i Etykiety kolumn odpowiadają kryteriom grupowania (jeśli stosujemy tylko jedno kryterium, obszar Etykiety wierszy lub Etykiety kolumn pozostaje pusty). W obszarze Wartości umieszczamy pole do obliczeń, wybieramy również odpowiednią funkcję (np. suma, średnia, licznik).

Podpunkt a

Aby rozwiązać podpunkt a), dodamy do zestawienia kolumnę *Miesiąc*, w której wyznaczymy miesiąc każdej sprzedaży, korzystając z funkcji wbudowanej (=MIESIĄC (C2)). Następnie tworzymy i sortujemy tabelę przestawną dla tabeli *sprzedaz*, grupującą dane wg miesięcy i sumującą liczbę zakupów w każdym miesiącu. Poniżej prezentujemy projekt układu dla tworzonej tabeli:

**Podpunkty b – d**

Ponieważ podpunkty b – d zadania wymagają powiązań między poszczególnymi tabelami, spróbujemy zbudować jedną tabelę kumulującą wszystkie te powiązania. W tym celu tworzymy kopię tabeli *sprzedaz* w nowym arkuszu (B) i dodajemy do niej kolumny: *Produkt_nazwa*, *Id_kategorii*, *Cena*, *Kategoria_nazwa*, *Wartość*. Poszczególne pola uzupełniamy korzystając z następujących zależności:

- wartości *Produkt_nazwa*, *Kategoria* i *Cena* można odczytać z tabeli produkt, w oparciu o *Id_produktu*,
- wartość *Kategoria_nazwa* można odczytać z tabeli kategoria, po wyznaczeniu *Id_kategorii*,
- pole *Wartość* jest równe iloczynowi pól *Ilosc* i *Cena*.

Docelowo chcielibyśmy uzyskać poniższy efekt:

	A	B	C	D	E	F	G	H	I	J	K
1	Id_zakupu	Id_produkt	Produkt_n	Data_zakupu	Ilosc	Miesiąc	Id_kategor	Cena	Kategoria	Wartość	
2	1	p34	940x16x7	2012-08-09	9	8	k6	2,89	paski_dyla	26,01	
3	2	p87	Oslonka_f	2012-08-07	12	8	k19	22,99	wyroby_ko	275,88	
4	3	p86	Oslonka_p	2012-08-03	26	8	k19	20,99	wyroby_ko	545,74	
5	4	p20	1_l_kontak	2012-01-05	2	1	k4	29,99	klej	59,98	
6	5	p33	940x16x5	2012-08-02	32	8	k6	2,19	paski_dyla	70,08	
7	6	p64	1000x700x	2012-04-19	14	4	k12	22,99	plyty_kork	321,86	
8	7	p48	40x60	2012-05-15	8	5	k10	25	tablice_ko	200	
9	8	p60	1000x700x	2012-05-30	1	5	k12	5,99	plyty_kork	5,99	
10	9	p58	kpl_12_mr	2012-05-28	1	5	k11	10,2	podkladki	10,2	
11	10	p88	Taca_pros	2012-07-16	5	7	k19	26,99	wyroby_ko	134,95	
12	11	p35	940x16x10	2012-06-12	26	6	k6	3,29	paski_dyla	85,54	
13	12	p91	Stozkowe	2012-06-28	14	6	k20	0,89	korki_do_l	12,46	
14	13	p30	940x23x5	2012-04-02	69	4	k6	2,19	paski_dyla	151,11	
15	14	p40	LK_3	2012-03-26	6	3	k8	3,6	listwy_kork	21,6	

Aby możliwe było utworzenia opisanej powyżej tabeli, potrzebna jest nam metoda na wyszukiwanie w tabeli wierszy z ustaloną wartością pewnego pola i wybieranie z takich wierszy wartości innych pól. Wykorzystaliśmy do tego funkcję *WYSZUKAJ.PIONOWO* o czterech parametrach:

- pierwszy określa wyszukiwaną wartość,
- drugi definiuje obszar przeszukiwania (wartość szukana jest w pierwszej kolumnie obszaru),
- trzeci parametr wskazuje, z której kolumny obszaru należy pobrać wynikową wartość,
- czwarty parametr określa czy wyszukiwanie ma być dokładne czy też przedziałowe (w naszym przypadku wpisujemy wartość *FALSZ* oznaczającą dokładne wyszukiwanie).

Działanie funkcji zilustrujemy na przykładzie. Załóżmy, że w kolumnie B znajdują się wartości *Id_produktu* z kolejnych wierszy tabeli *sprzedaz*. Aby w kolumnie C uzyskać nazwę produktu, wpisujemy następujące parametry funkcji *WYSZUKAJ.PIONOWO* (w wierszu 2):

- B2 jako wyszukiwaną wartość;
- obszar zajmowany przez tabelę produktów (w naszym przykładzie *produkt!\$A\$2:\$E\$100*); stosujemy adresowanie bezpośrednie, aby obszar nie zmieniał się przy kopiowaniu;
- liczbę 2 jako trzeci parametr, gdyż nazwy produktów znajdują się w drugiej kolumnie tabeli produkt.

Poniżej załączamy ilustrację tego przykładu:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Id_zakupu	Id_produkt	Produkt_n	Data_zakupu	Ilosc	Miesiąc	Id_kategor	Cena	Kategoria	Wartość		
2		p34	=WYSZUKAJ.PIONOWO(B2;produkt!\$A\$2:\$E\$100;2;FALSZ)						paski_dyla	26,01		
3	2	p87	Oslonka_f	2012-08-07	12	8	k19	22,99	wyroby_ko	275,88		
4	3	p86	Oslonka_p	2012-08-03	26	8	k19	20,99	wyroby_ko	545,74		
5	4	p20	1_l_kontak	2012-01-05	2	1	k4	29,99	klej	59,98		
6	5	p33	940x16x5	2012-08-02	32	8	k6	2,19	paski_dyla	70,08		
7	6	p64	1000x700x	2012-04-19	14	4	k12	22,99	plyty_kork	321,86		
8	7	p48	40x60	2012-05-15	8	5	k10	25	tablice_ko	200		
9	8	p60	1000x700x	2012-05-30	1	5	k12	5,99	plyty_kork	5,99		
10	9	p58	kpl_12_mr	2012-05-28	1	5	k11	10,2	podkladki	10,2		
11	10	p88	Taca_pros	2012-07-16	5	7	k19	26,99	wyroby_ko	134,95		
12	11	p35	940x16x10	2012-06-12	26	6	k6	3,29	paski_dyla	85,54		
13	12	p91	Stozkowe	2012-06-28	14	6	k20	0,89	korki_do_l	12,46		
14	13	p30	940x23x5	2012-04-02	69	4	k6	2,19	paski_dyla	151,11		
15	14	p40	LK_3	2012-03-26	6	3	k8	3,6	listwy_kork	21,6		

Mając dostęp do tabeli, w której wiersze z tabeli sprzedaz uzupełnione są o powiązane z nimi informacje z tabel produkt i kategoria, punkty b), c) i d), możemy rozwiązać, stosując tabele przestawne w powiązaniu z sortowaniem i filtrowaniem danych.

Podpunkt b

Stosujemy grupowanie wg nazwy kategorii przeciągając *Kategoria_nazwa* jako pole wierszy.

W obszarze wartości umieszczamy sumę pola *Wartość* (przeciągamy nazwę pola do odpowiedniego obszaru).

Podpunkt c

Sortujemy dane malejąco względem nazwy kategorii, co spowoduje, że „wyroby_korkowe” pojawią się w początkowym bloku tabeli. Tabelę przestawną tworzymy tylko dla tego fragmentu tabeli. Wybieramy *Produkt_nazwa* jako pole wierszy, a w obszarze wartości umieszczamy sumę pola *Ilość* (i wg niej sortujemy).

Podpunkt d

Aby ograniczyć analizę do podanych kategorii, najpierw sortujemy dane wg nazwy kategorii i usuwamy wiersze z innych kategorii niż *parkiet_korkowy* i *panele_korkowe*. Za pole wierszy przyjmujemy *Produkt_nazwa*, pole kolumn to *Kategoria_nazwa*, a w obszarze wartości umieszczamy sumę pola *Ilość*.

Zapalonym programistom zalecić warto lepsze poznanie języka *SQL*, w którym można samodzielnie formułować zapytania do bazy danych lub modyfikować zapytania utworzone za pomocą interaktywnych narzędzi do tworzenia kwerend. Pozwoli to rozwiązywać zadania tego typu z wykorzystaniem bardzo elastycznego języka, unikając ograniczeń bądź niedogodności interaktywnego tworzenia kwerend w MS Access.

Rozwiązania bazodanowe dla podpunktów a) – c) uzyskać można, tworząc dość standardowe kwerendy wybierające z zastosowaniem grupowania. Po utworzeniu relacji łączących wszystkie trzy tabele, punkty te nie powinny nastręczać większych trudności wprawnym użytkownikom MS Access lub innych systemów bazodanowych. W rozwiązaniu podpunktu d) zaprezentowaliśmy potencjał kwerend krzyżowych, choć punkt ten można również rozwiązać w sposób bardziej standardowy, korzystając z faktu, że wynik należy uzyskać tylko dla dwóch kategorii produktów (można na przykład utworzyć dwie kwerendy – jedną dla kategorii *parkiet_korkowy*, a drugą dla kategorii *panele_korkowe*).

Odnosnie zaprezentowanego rozwiązania zadania w arkuszu kalkulacyjnym należy zaznaczyć, że wiązanie tabel poprzez funkcję *WYSZUKAJ. PIONOWO* jest nieefektywne dla dużych zbiorów danych oraz niewygodne przy aktualizacji danych. Zaprezentowaliśmy to rozwiązanie, aby podkreślić uniwersalność różnych narzędzi i zwrócić uwagę, że maturzysta samodzielnie podejmuje decyzję o wyborze narzędzia do rozwiązania zadania (uwzględniając czas dostępny w trakcie egzaminu maturalnego).

Zadanie 20. Hotel Panorama (0–10)

Dane są trzy pliki tekstowe o nazwach: `klienci.txt`; `pokoje.txt`; `noclegi.txt`. Zawierają one informacje na temat zrealizowanych usług hotelowych hotelu Panorama w okresie wakacyjnym, tzn. zameldowanie gościa nastąpiło od 1.07.2013 do 31.08.2013.

Pierwszy wiersz każdego z plików jest wierszem nagłówkowym, a dane w wierszach rozdzielone są znakami tabulacji.

Plik o nazwie `klienci.txt` zawiera informacje o gościach hotelu. W każdym wierszu pliku znajdują się następujące dane: numer dowodu osobistego gościa hotelu (`nr_dowodu`), nazwisko (`nazwisko`), imię (`imie`) i miejsce zamieszkania (`miestowosc`).

Przykład:

<code>nr_dowodu</code>	<code>nazwisko</code>	<code>imie</code>	<code>miestowosc</code>
SAS253401	Pastuszek	Joanna	Szczecin
UNC608098	Siudut	Anna	Jaworzno
NMZ567271	Konopka	Kamil	Tarnowskie Gory

Plik o nazwie `pokoje.txt` zawiera w każdym wierszu: numer pokoju (`nr_pokoju`), maksymalną liczbę osób, które mogą nocować w pokoju (`liczba_osob`), standard pokoju (`standard`), gdzie S – oznacza pokój standardowy, W – pokój o podwyższonym standardzie oraz cenę wynajęcia pokoju na jedną dobę (`cena`).

Przykład:

<code>nr_pokoju</code>	<code>liczba_osob</code>	<code>standard</code>	<code>cena</code>
101	2	S	220
102	2	S	220
103	2	S	220

Plik o nazwie `noclegi.txt` zawiera w każdym wierszu: identyfikator noclegu (`id_noc`), datę przyjazdu gościa hotelu (`data_przyjazdu`), datę wyjazdu gościa (`data_wyjazdu`), numer dowodu osobistego gościa korzystającego z danego noclegu (`nr_dowodu`), numer pokoju (`nr_pokoju`), oraz dodatkowe usługi, z których skorzystał gość podczas noclegu (`uslugi`).

Przykład:

<code>id_noc</code>	<code>data_przyjazdu</code>	<code>data_wyjazdu</code>	<code>nr_dowodu</code>	<code>nr_pokoju</code>	<code>uslugi</code>
198	2013-07-10	2013-07-12	JAA932190	501	2020
199	2013-07-10	2013-07-11	SIS395155	108	1010
206	2013-07-10	2013-07-13	RMS452742	113	0030

Korzystanie z dodatkowych usług odnotowane jest w postaci liczb jednocyfrowych. Na pierwszej pozycji od lewej strony odnotowana jest liczba konsumowanych śniadań (cena jednostkowa 20 zł), na drugiej pozycji – liczba wejść na basen (cena jednostkowa 30 zł), na trzeciej – parking (cena jednostkowa 15 zł), a na ostatniej pozycji – czyszczenie ubrania (cena jednostkowa 35 zł).

Na przykład zapis:

2020 – oznacza, że gość 2 razy jadł śniadanie w hotelu (2*20 zł), nie korzystał z basenu, przez 2 doby parkował samochód na hotelowym parkingu (2*15 zł) oraz nie oddawał ubrania do czyszczenia. W związku z tym do kosztu noclegu dopisuje się koszt dodatkowych usług w wysokości 70 zł.

1130 – oznacza, że gość zjadł jeden raz śniadanie w hotelu (1*20 zł), jeden raz skorzystał z basenu (1*30 zł) i przez 3 doby parkował samochód na hotelowym parkingu (3*15 zł) oraz nie oddawał ubrania do czyszczenia. Gościowi do kosztu noclegu dopisuje się koszt dodatkowych usług w wysokości 95 zł.

Uwaga: Żadna z usług nie była zamawiana więcej niż 9 razy.

Korzystając z danych zawartych w tych plikach oraz z dostępnych narzędzi informatycznych, wykonaj poniższe polecenia. Każdą odpowiedź umieść w pliku `wyniki.txt`, poprzedzając ją oznaczeniem odpowiedniego podpunktu od a) do e).

- Podaj imię i nazwisko gościa, który skorzystał z największej liczby noclegów podczas jednorazowego pobytu w hotelu Panorama. Podaj liczbę tych noclegów.
- Podaj zestawienie (imię i nazwisko) osób z Krakowa goszczących w hotelu Panorama, którzy korzystali z parkingu.
- Utwórz listę miejscowości, z których co najmniej 15 **różnych** osób nocowało choć raz w hotelu Panorama. Zestawienie posortuj alfabetycznie.
- Podaj kwotę uzyskaną z tytułu wynajmu pokoi oraz kwotę uzyskaną z opłat za korzystanie przez gości z dodatkowych usług w okresie od 1.07.2013 do 31.08.2013.
- Podaj numery pokoi o podwyższonym standardzie, z których nigdy **nie korzystali** goście z Krakowa.

Do oceny oddajesz plik(i) o nazwie(ach)
tu wpisz nazwę(y) pliku(ów)
 zawierający(e) komputerową(e) realizację(e) Twoich obliczeń oraz plik tekstowy o nazwie `wyniki.txt` z odpowiedziami do podpunktów a), b), c), d), e).

Wymagania ogólne	<i>II. Wyszukiwanie, gromadzenie i przetwarzanie informacji z różnych źródeł; opracowywanie za pomocą komputera: rysunków, tekstów danych liczbowych, motywów, animacji, prezentacji multimedialnych.</i>
Wymagania szczegółowe	<i>2. Wyszukiwanie, gromadzenie, selekcjonowanie, przetwarzanie i wykorzystywanie informacji, korzystanie z różnych źródeł i sposobów zdobywania informacji.</i> <i>Zdający:</i> <i>2) projektuje relacyjną bazę danych z zapewnieniem integralności danych,</i> <i>3) stosuje metody wyszukiwania i przetwarzania informacji w relacyjnej bazie danych; tworzy aplikację bazodanową, wykorzystującą język zapytań, kwerendy, raporty; zapewnia integralność danych na poziomie pól, tabel, relacji.</i>

Schemat punktowania

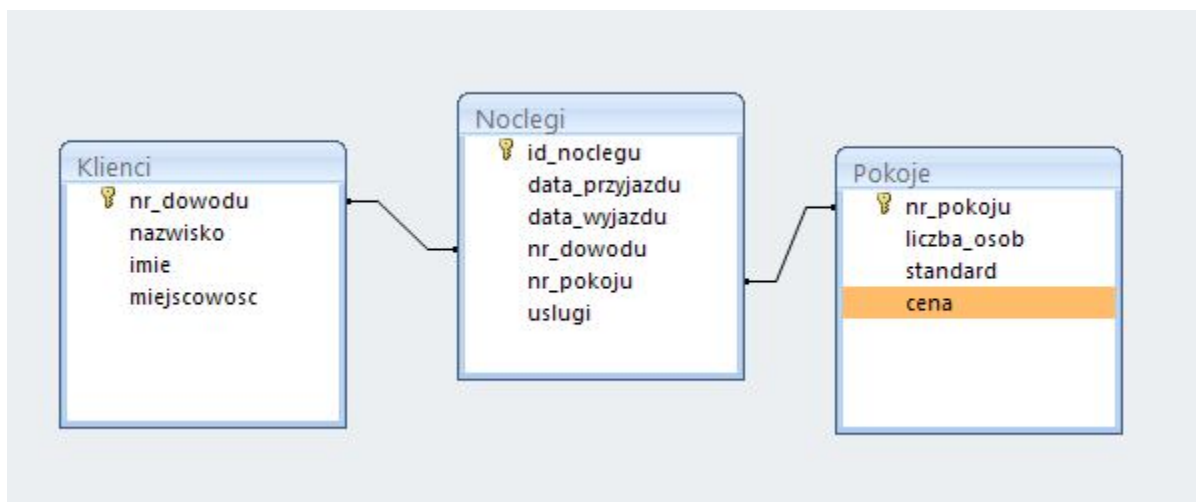
Podpunkt	Czynność	Liczba punktów za podpunkt	Liczba punktów za zadanie
a	Za poprawną odpowiedź – 2 punkty, w tym za: • imię i nazwisko – 1 punkt. • liczbę noclegów – 1 punkt.	2	10
b	Za poprawną odpowiedź – 2 punkty. W przypadku powtórzeń – 1 punkt.	2	
c	Za poprawną odpowiedź – 2 punkty, w tym za: • podanie listy – 1 punkt. • za prawidłowe posortowanie – 1 punkt.	2	
d	Za poprawną odpowiedź – 2 punkty, w tym za: • kwotę uzyskaną z tytułu wynajmu pokoi – 1 punkt. • kwotę uzyskaną z opłat za korzystania przez gości z dodatkowych usług – 1 punkt.	2	
e	Za poprawną odpowiedź – 2 punkty. W przypadku braku jednego numeru lub wystąpienia jednego numeru nieprawidłowego – 1 punkt.	2	

Zadanie 20. Hotel Panorama (0–10) – rozwiązanie

Plik zawierający rozwiązanie znajdują się w folderze *HOTEL*.

Komentarz

Zadanie Hotel Panorama należy do grupy zadań, które mogą być rozwiązywane narzędziami bazodanowymi lub z wykorzystaniem arkusza kalkulacyjnego. Baza składa się z trzech tabel: klienci (klucz główny – nr_dowodu), pokoje (klucz główny – nr_pokoju) i noclegi (klucz główny – Id_noclegu). Tabela klienci połączona zostanie z tabelą noclegi relacją typu „jeden do wielu”, podobnie tabelę pokoje z tabelą noclegi łączy relacja typu „jeden do wielu”.



W Accessie import plików z danymi jest prosty. Ambitni maturzyści mogą utworzyć bazę i zapytania w języku SQL:

```
-- utworzenie bazy danych hotel
create database hotel;
```

```
-- ustawienie bazy hotel jako domyślnej
use hotel;
```

```
-- utworzenie tabeli noclegi
CREATE TABLE `noclegi` (
  `id_noc` int NOT NULL,
  `data_przyjazdu` date NOT NULL,
  `data_wyjazdu` date NOT NULL,
  `nr_dowodu` char(9) NOT NULL,
  `nr_pokoju` int NOT NULL,
  `uslugi` int(1) NOT NULL,
  PRIMARY KEY (`id_noc`)
);
```

```
-- załadowanie danych do tabeli noclegi
LOAD DATA INFILE
"C:\Dane_hotel\noclegi.csv" INTO TABLE `noclegi` FIELDS
TERMINATED BY ',' ENCLOSED BY '"' ESCAPED BY '\\' LINES
TERMINATED BY '\n' (id_noc,data_przyjazdu,data_wyjazdu,nr_dowodu,nr_pokoju,uslugi);
```

```
-- utworzenie tabeli klienci
CREATE TABLE `klienci` (
  `nr_dowodu` char(9) NOT NULL,
  `nazwisko` char(20) NOT NULL,
  `imie` char(20) NOT NULL,
  `miejscowosc` char(20) NOT NULL,
  PRIMARY KEY (`nr_dowodu`)
);
```

```
-- załadowanie danych do tabeli klienci
LOAD DATA INFILE
"C:\Dane_hotel\klienci.csv" INTO TABLE `klienci` FIELDS
TERMINATED BY ',' ENCLOSED BY '"' ESCAPED BY '\\' LINES
TERMINATED BY '\n' (nr_dowodu,nazwisko,imie,miejscowosc);
```

```
-- utworzenie tabeli pokoje
CREATE TABLE `pokoje` (
  `nr_pokoju` int(11) NOT NULL,
  `liczba_osob` int(11) NOT NULL,
  `standard` char(1) NOT NULL,
  `cena` int(11) NOT NULL,
  PRIMARY KEY (`nr_pokoju`)
);
```

-- załadowanie danych do tabeli pokoje

LOAD DATA INFILE

"C:\Dane_hotel\pokoje.csv" INTO TABLE `pokoje` FIELDS

TERMINATED BY ',' ENCLOSED BY '"' ESCAPED BY '\\' LINES

TERMINATED BY '\n' (nr_pokoju,liczba_osob,standard,cena);

W poleceniu a) należy podać imię i nazwisko gościa, który skorzystał z największej liczby noclegów podczas jednorazowego pobytu w hotelu Panorama oraz liczbę tych noclegów. Aby podać odpowiedź na tak postawione pytanie, należy obliczyć liczbę nocy, odejmując od daty wyjazdu datę przyjazdu, znaleźć maksimum i odpowiadającą temu maksimum osobę.

Zapytanie a:

```
SELECT Klienci.imie, Klienci.nazwisko, (data_wyjazdu-data_przyjazdu) AS Liczba_nocy
FROM Klienci, Noclegi
```

```
WHERE ((Klienci.nr_dowodu)=(Noclegi.nr_dowodu))
```

```
ORDER BY (data_wyjazdu-data_przyjazdu) DESC;
```

W MySQLu do obliczenia liczby noclegów należy użyć funkcji TO_DAYS:

```
SELECT imie, nazwisko, to_days(data_wyjazdu)-to_days(data_przyjazdu) AS liczba_nocy
FROM Klienci, Noclegi
```

```
WHERE ((Klienci.nr_dowodu)=(Noclegi.nr_dowodu))
```

```
ORDER BY liczba_nocy DESC;
```

Aby wykonać zestawienie gości z Krakowa, którzy korzystali z parkingu, należy utworzyć zapytanie odnoszące się do dwóch tabel: Klienci i Noclegi. Informacja o korzystaniu gościa z parkingu zapisana została w kolumnie „usługi” na trzeciej pozycji, licząc od strony lewej. Cyfra oznaczająca liczbę parkowań powinna być różna od zera. Jeżeli kolumna „usługi” przechowuje dane typu char, to wyodrębnienie informacji o parkowaniu można uzyskać za pomocą funkcji MID(nazwa_kolumny,pierwszy_znak,liczba_znaków).

Jeżeli ktoś preferuje pracę w arkuszu kalkulacyjnym, może zastosować funkcję FRAGMENT.TEKSTU(tekst,liczba_początkowa,liczba_znaków). Dodatkowym warunkiem jest pochodzenie gości z Krakowa (tabela Klienci, kolumna miejscowość).

Zapytanie b:

```
SELECT klienci.nr_dowodu, imie, nazwisko
```

```
FROM Klienci, Noclegi
```

```
WHERE Klienci.nr_dowodu = Noclegi.nr_dowodu AND Miejscowosc='Krakow' AND
```

```
Mid(uslugi,3,1)<>'0'
```

```
GROUP BY klienci.nr_dowodu, imie, nazwisko;
```

W poleceniu SELECT obok imienia i nazwiska dodano pole nr_dowodu, które jednoznacznie identyfikuje klienta, aby wykluczyć pominięcie osób o takim samym imieniu i nazwisku.

W celu utworzenia listy miejscowości, z których co najmniej 15 **różnych** osób nocowało choć raz w hotelu Panorama, należy utworzyć zapytanie pomocnicze w wyniku którego otrzymamy zestawienie numerów dowodów osobistych (z tabeli Noclegi) oraz miejscowości (z tabeli Klienci). Dzięki grupowaniu otrzymamy zestawienie bez powtórzeń (czyli noclegi różnych osób).

Zapytanie c:

```
SELECT Noclegi.nr_dowodu, Klienci.miejscowosc
```

```
FROM Noclegi, Klienci
```

```
WHERE ((Noclegi.nr_dowodu)=(Klienci.nr_dowodu))
```

```
GROUP BY Noclegi.nr_dowodu, Klienci.miejscowosc;
```

Korzystając z wyników pomocniczego zapytania, policzymy miejscowości i wypiszemy tylko te, które występowały w zestawieniu pomocniczym przynajmniej 15 razy (`HAVING count(*)>=15`). Na koniec należy je uporządkować alfabetycznie (`ORDER BY` miejscowość).

Kwerenda c1:

```
SELECT miejscowosc,
FROM c
GROUP BY miejscowosc
HAVING count(*)>=15
ORDER BY miejscowosc;
```

Aby podać kwotę uzyskaną z tytułu wynajmu pokoi, należy w pierwszej kolejności dla każdego identyfikatora noclegu obliczyć iloczyn liczby nocy i ceny pojedynczego noclegu w wynajmowanym pokoju.

Kwerenda d:

```
SELECT id_noc, (to_days(data_wyjazdu)-to_days(data_przyjazdu))*cena AS cena_noclegow
FROM Pokoje, Noclegi
WHERE Pokoje.nr_pokoju=Noclegi.nr_pokoju;
```

Po zsumowaniu wartości wszystkich noclegów otrzymamy kwotę uzyskaną z tytułu wynajmu pokoi:

Kwerenda d1:

```
SELECT Sum(cena_noclegow) AS Za_noclegi
FROM d;
```

Aby obliczyć kwotę uzyskaną z opłat za korzystanie przez gości z dodatkowych usług, należy dla każdego identyfikatora noclegu wyodrębnić liczbę wykorzystanych usług i pomnożyć przez cenę tej usługi. Funkcja `VAL` zamienia wartość typu `char` na wartość typu `int`. W MySQLu będzie to funkcja `CAST`.

Kwerenda d2:

```
SELECT Sum(20*Val(Mid(uslugi,1,1))) AS sniadanie,
Sum(30*Val(Mid(uslugi,2,1))) AS basen,
Sum(15*Val(Mid(uslugi,3,1))) AS parking,
Sum(35*Val(Mid(uslugi,4,1))) AS czyszczenie
FROM Noclegi;
```

Kwerenda d2:

```
SELECT sniadanie+basen+parking+czyszczenie AS Za_uslugi
FROM d2;
```

W ostatnim poleceniu należy podać numery pokoi o podwyższonym standardzie, z których nigdy **nie korzystali** goście z Krakowa. W tym celu zostało utworzone podzapytanie dające w wyniku numery pokoi o podwyższonym standardzie w których byli goście z Krakowa, a następnie pytanie zewnętrzne: wybierz numery pokoi o podwyższonym standardzie z tabeli pokoje nie występujące w zestawieniu będącym wynikiem podzapytania.

Kwerenda e:

```
SELECT Pokoje.nr_pokoju
FROM pokoje
WHERE standard='W' AND pokoje.nr_pokoju NOT IN
(SELECT Noclegi.nr_pokoju
FROM Noclegi, Pokoje, Klienci
```

```
WHERE Klienci.nr_dowodu = Noclegi.nr_dowodu AND Noclegi.nr_pokoju =
Pokoje.nr_pokoju
AND standard='W' AND miejscowosc='krakow'
GROUP BY Noclegi.nr_pokoju);
```

Zadanie 21. Podzielność (0–10)

W trzech plikach tekstowych `liczby1.txt`, `liczby2.txt` i `liczby3.txt` zapisano po 1000 dodatnich liczb binarnych. W każdym pliku liczby zapisano w kolejnych wierszach po jednej liczbie w wierszu. W pliku `liczby1.txt` długość zapisu każdej z liczb jest nie większa od 12. W pliku `liczby2.txt` długość zapisu każdej z liczb jest nie większa od 30, zaś w pliku `liczby3.txt` długość zapisu każdej liczby nie przekracza 200.

Dla każdego z plików z danymi wyznacz, ile zawiera on

- liczb podzielnych przez 2,
- liczb podzielnych przez 3,
- liczb podzielnych przez 5.

Przykład:

W pliku z 3 liczbami binarnymi:

10101

1100

1110

są 2 liczby podzielne przez 2, 1 liczba podzielna przez 3 i 1 liczba podzielna przez 5.

Do oceny oddajesz plik(i) o nazwiezawierający

tu wpisz nazwę pliku/plików

komputerową realizację Twoich obliczeń oraz plik tekstowy `podzielnosc.txt` zawierający w dziewięciu kolejnych wierszach dziewięć liczb, po jednej w wierszu. Pierwsze trzy wiersze powinny zawierać liczby liczb z pliku `liczby1.txt` podzielnych odpowiednio przez 2, 3 i 5. Kolejne trzy wiersze powinny zawierać liczby liczb z pliku `liczby2.txt` podzielnych odpowiednio przez 2, 3 i 5, a ostatnie trzy wiersze liczby liczb z pliku `liczby3.txt` podzielnych odpowiednio przez 2, 3 i 5.

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin,</i> <i>11) opisuje podstawowe algorytmy i stosuje algorytmy na liczbach całkowitych,</i> <i>12) projektuje rozwiązanie problemu (realizację algorytmu) i dobiera odpowiednią strukturę danych,</i> <i>15) stosuje zasady programowania strukturalnego i modularnego do rozwiązywania problemu,</i> <i>23) stosuje podstawowe konstrukcje programistyczne w wybranym języku programowania, instrukcje iteracyjne i warunkowe, rekurencję i procedury,</i>

	instrukcje wejścia i wyjścia, poprawnie tworzy strukturę programu, 24) dobiera najlepszy algorytm, odpowiednie struktury danych i oprogramowanie do rozwiązania postawionego problemu, 26) ocenia poprawność komputerowego rozwiązania na podstawie jego testowania.
--	---

Schemat punktowania

Czynność	Liczba punktów za zadanie
Za poprawne wyniki dla pliku 1 – 2 punkty. Za poprawne wyniki dla pliku 2 – 3 punkty. Za poprawne wyniki dla pliku 3 – 4 punkty.	9

Zadanie 21. Podzielność (0–10) - rozwiązanie

Pliki z danymi, plik programu źródłowego oraz plik `podzielnosc.txt` zawierający odpowiedzi znajdują się w folderze `PODZIELNOSC`.

Komentarz

Jedna z metod rozwiązania tego zadania mogła by polegać na przytoczeniu i wykorzystaniu własności podzielności liczb binarnych. O ile własność podzielności przez 2 jest oczywista i powszechnie znana – najmniej znaczącą cyfrą takiej liczby musi być 0 – to już własności podzielności liczby binarnych przez inne liczby nie są tak naturalne i znane. Dla każdego dzielnika taka własność byłaby pewnie inna i prowadziłaby do algorytmu właściwego tylko dla tej własności. Przedstawimy algorytm, który jest jednakowy dla wszystkich dzielników z dokładnością do parametru, którym jest właśnie dzielnik.

Jest naturalne rozwiązanie, które pozwala stwierdzić, czy dana, dodatnia liczba binarna b , jest podzielna całkowicie przez dodatnią (dziesiętną) liczbę całkowitą p . Wystarczy obliczyć (dziesiętną) wartość d liczby b i sprawdzić, czy p dzieli całkowicie d . W tym celu wykorzystujemy dostępny w większości języków programowania operator **mod** obliczania reszty z dzielenia liczb całkowitych i sprawdzamy, czy $d \bmod p = 0$. Do zamiany liczby binarnej b na jej odpowiednik dziesiętny d można zastosować schemat Hornera. Prawda jakie to proste? Jest tylko jeden mały problem. W kolejnych plikach z danymi są coraz większe liczby. W pliku `liczby1.txt` długość zapisu każdej z liczb jest nie większa niż 12 i do zapisu każdej takiej liczby wystarczają dwa bajty, do zapisu jednej liczby z pliku `liczby2.txt` potrzeba już 4 bajtów, natomiast do zapisu liczby z pliku `liczby3.txt` może być potrzebnych 25 bajtów na każdą z nich. Jeśli język programowania użyty do rozwiązania pozwala reprezentować tak duże liczby i umożliwia wykonywanie na nich podstawowych arytmetycznych – dodawanie, mnożenie i branie modulo – to powyżej opisane rozwiązanie jest wystarczające. Często jednak języki programowania nie umożliwiają bezpośredniego operowania na bardzo dużych liczbach. Wówczas pozostaje albo zaprogramować własną arytmetykę dużych liczb, albo, jak w tym przypadku, skorzystać z pewnych własności używanych operacji arytmetycznych. Tutaj skorzystamy z własności operacji modulo brania reszty z dzielenia, które to własności pozwalają nam operować tylko na resztach z dzielenia przez dzielnik p , a nie na całych liczbach. Każda taka reszta jest nie większa od p . Dwie podstawowe, wykorzystywane przez nas własności są następujące:

Dla dodatnich liczb całkowitych a , b i p mamy:

- (1) $(a + b) \bmod p = (a \bmod p + b \bmod p) \bmod p$
- (2) $(a * b) \bmod p = ((a \bmod p) * (b \bmod p)) \bmod p$

Tak więc do policzenia reszty z dzielenia przez p dodatniej liczby całkowitej d , której wartość jest taka sama jak wartość liczby binarnej b , należy po prostu zastosować schemat Hornera obliczania wartości d z zapisu b pamiętając, żeby wszystkie obliczenia wykonywać modulo p . Oto funkcja zapisana w języku programowania C++, która dla liczby binarnej b podanej w postaci zero-jedynkowego napisu i dodatniego, całkowitego podzielnika p , oblicza resztę z dzielenia wartości dziesiętnej liczby b przez p .

```
int Reszta(string b, int p){
    const char zero = '0';

    int dl = b.length(); //obliczenie długości zapisu liczby b
    int d = 0; // po zakończeniu obliczeń wartością d będzie
    // wartość dziesiętna liczby b modulo p

    // schemat Hornera modulo p
    for (int j = 0; j < dl; j++){
        int cyfra = b[j] - zero; //odzyskanie kolejnej cyfry liczby b
        // poczynając od najbardziej znaczącej
        d = (d*2 + cyfra) % p; // % jest operatorem modulo w C++
    }
    return d;
}
```

Należy zwrócić jeszcze uwagę na drobiazg, jakim jest odzyskiwanie wartości liczbowych kolejnych cyfr zapisu liczby b . Liczba b jest zadana jako napis złożony ze znaków '0' i '1'. Każdy ze znaków '0' lub '1' do obliczeń arytmetycznych należy zamienić odpowiednio na liczbę 0 lub 1. Do tego celu najprościej wykorzystać fakt, że cyfry '0', '1', ..., '9' są kodowane kolejnymi liczbami naturalnymi poczynając od kodu znaku '0'. Tak więc, gdy od kodu cyfry odejmiemy kod cyfry '0', to otrzymamy liczbę odpowiadającą tej cyfrze. W przedstawionym powyżej rozwiązaniu wykorzystujemy ten fakt – wartość wyrażenia $b[j] - \text{zero}$ jest obliczana z wykorzystaniem kodów cyfr $b[j]$ i zero .

Poniżej przedstawiamy program *Podzielnosc* zapisany w języku C++, który oblicza żądane wyniki dla jednego pliku. Dla przykładu, żeby uzyskać wyniki dla pliku *liczby1.txt* można program wykonywalny *Podzielnosc* uruchomić w katalogu zawierającym plik *liczby1.txt* następującą komendą z wiersza poleceń:

```
Podzielnosc < liczby1.txt > wyniki1.txt
```

Wyniki znajdują się w pliku tekstowym *wyniki1.txt* w tym samym katalogu. Na koniec trzeba pamiętać, żeby wyniki obliczeń dla wszystkich trzech plików umieścić w jednym pliku tekstowym *podzielnosc.txt* zgodnie z opisem w treści zadania.

```
//Program Podzielnosc
#include <iostream>
using namespace std;

int Reszta(string b, int p){
    const char zero = '0';
```

```
int dl = b.length(); //obliczenie długości zapisu liczby b
int d = 0; //po zakończeniu obliczeń wartością d będzie
// wartość dziesiętna liczby b modulo p

// schemat Hornera modulo p
for (int j = 0; j < dl; j++){
    int cyfra = b[j] - zero; //odzyskanie kolejnej cyfry liczby
    b
    //poczynając od najbardziej znaczącej
    d = (d*2 + cyfra) % p; //% jest operatorem modulo w C++
}
return d;
}
int main(){
string liczba;
const int n = 1000; //rozmiar danych
int podz_2 = 0, podz_3 = 0, podz_5 = 0; // podz_p - liczba
liczb
// podzielnych przez p
for (int i = 0; i < n; i++){
    cin >> liczba; //wczytanie kolejnej liczby;
    if (Reszta(liczba,2) == 0) podz_2++;
    if (Reszta(liczba,3) == 0) podz_3++;
    if (Reszta(liczba,5) == 0) podz_5++;
}
cout << "podzielne przez " << 2 << " : " << podz_2 << "\n";
cout << "podzielne przez " << 3 << " : " << podz_3 << "\n";
cout << "podzielne przez " << 5 << " : " << podz_5 << "\n";

return 0;
}
```

Zadanie 22. Bloki trójkowe (0–12)

Niech n będzie dodatnią liczbą całkowitą i niech $a_1, a_2, \dots, a_n$ będzie ciągiem nieujemnych liczb całkowitych. Dla pary liczb i, j takich, że $1 \leq i \leq j \leq n$, **blokiem** $b(i, j)$ nazywamy podciąg kolejnych elementów ciągu a z pozycji od i do j , czyli $a_i, a_{i+1}, \dots, a_j$. **Długością bloku** nazywamy liczbę jego elementów. O bloku, którego suma elementów jest podzielna przez 3 mówimy, że jest **blokiem trójkowym**.

Przykład:

W ciągu 0,0,2,3,2,1,2 najdłuższym blokiem trójkowym jest $b(4,6) = 3,2,1$.

W plikach tekstowych `bloki1.txt`, `bloki2.txt` i `bloki3.txt` zapisano ciągi odpowiednio 1000, 30000 i 1000000 nieujemnych liczb całkowitych mniejszych od 10 000. W każdym pliku liczby zapisano w kolejnych wierszach, po jednej liczbie w każdym wierszu.

Dla każdego pliku z danymi wyznacz **długość najdłuższego bloku trójkowego** w ciągu zapisanym w tym pliku.

Przykład:

Dla danych z pliku z 7 liczbami:

0
0
2
3
2
1
2

długość najdłuższego bloku trójkowego wynosi 3.

Do oceny oddajesz plik(i) o nazwie(ach)

tu wpisz nazwę/nazwy pliku/plików

zawierający(e) komputerową realizację Twoich obliczeń oraz pliki tekstowe `wyniki1.txt`, `wyniki2.txt`, `wyniki3.txt`, gdzie każdy z nich zawiera liczbę równą długości najdłuższego bloku trójkowego w ciągach zapisanych odpowiednio w plikach `bloki1.txt`, `bloki2.txt` i `bloki3.txt`.

Wymagania ogólne	<i>III. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, z zastosowaniem podejścia algorytmicznego.</i>
Wymagania szczegółowe	<i>5. Rozwiązywanie problemów i podejmowanie decyzji z wykorzystaniem komputera, stosowanie podejścia algorytmicznego.</i> <i>Zdający:</i> <i>1) analizuje, modeluje i rozwiązuje sytuacje problemowe z różnych dziedzin,</i> <i>11) opisuje podstawowe algorytmy i stosuje algorytmy na liczbach całkowitych,</i> <i>14) dobiera odpowiednie struktury danych do realizacji algorytmu, w tym struktury dynamiczne,</i> <i>15) stosuje zasady programowania strukturalnego i modularnego do rozwiązywania problemu,</i>

	23) stosuje podstawowe konstrukcje programistyczne w wybranym języku programowania, instrukcje iteracyjne i warunkowe, rekurencję i procedury, instrukcje wejścia i wyjścia, poprawnie tworzy strukturę programu, 24) dobiera najlepszy algorytm, odpowiednie struktury danych i oprogramowanie do rozwiązania postawionego problemu, 26) ocenia poprawność komputerowego rozwiązania na podstawie jego testowania.
--	--

Schemat punktowania

Czynność	Liczba punktów za zadanie
Za poprawne wyniki dla pliku 1 – 3 punkty. Za poprawne wyniki dla pliku 2 – 4 punkty. Za poprawne wyniki dla pliku 3 – 5 punktów.	12

Zadanie 22. Bloki trójkowe (0–12) – rozwiązanie

Pliki z danymi, plik programu źródłowego oraz pliki wynikowe zawierające odpowiedzi znajdują się w folderze BLOKI TROJKOWE.

Komentarz

Uważny czytelnik szybko zauważy, że proste rozwiązanie jest ukryte w treści zadania. Wystarczy przejrzeć wszystkie bloki, dla każdego bloku zsumować jego elementy, sprawdzić, czy otrzymana suma jest podzielna przez 3 i spośród wszystkich bloków spełniających ten warunek wybrać najdłuższy. Oto fragment algorytmu zapisany w pseudo-języku C++, będący realizacją tego prostego pomysłu:

```
najdluzszy = 0; // długość najdłuższego z dotychczas
                //przejrzanych bloków trójkowych
for (i = 1; i <= n; i++)
  for (j = i; j <= n; j++){
    // obliczamy sumę elementów w bloku b(i,j)
    suma = 0;
    for (k = i; k <= j; k++)
      suma = suma + ak; // (*)
    // sprawdzamy, czy suma jest podzielna przez 3, czyli
    // czy reszta z dzielenia suma przez 3 daje 0
    if (suma % 3 == 0){
      // jeśli suma jest podzielna przez 3 i blok b(i,j)
      // jest dłuższy od dotychczas najdłuższego bloku
      // trójkowego, to zapamiętujemy jego długość
      dl = j - i + 1;
      if (dl >= najdluzszy)
        najdluzszy = dl;
    }
  }

// najdluzszy jest długością najdłuższego bloku trójkowego w
// ciągu a
```

Jedyną dobrą cechą powyższego rozwiązania jest jego prostota i to, że w ogóle mamy jakiekolwiek rozwiązanie. Poza tym przedstawione rozwiązanie ma same wady.

*Po pierwsze musimy pamiętać o tym, żeby arytmetyka języka programowania, którego używamy, umożliwiała operowanie na liczbach pojawiających się w obliczeniach. Zauważmy, że największa liczba jaka mogłaby się w obliczeniach dla opisanych danych wynosi $1\,000\,000 * 9\,999 = 9\,999\,000\,000$. W języku programowania C++ największa liczba typu `int` ze znakiem ma wartość 2 147 483 647, a bez znaku 4 294 967 295. Tak jest, gdy typ `int` jest 32-bitowy. Dla 16-bitowego typu `int` odpowiednie wielkości są znacząco mniejsze. Oczywiście można by użyć typów pozwalających na operowanie na dużo większych liczbach, ale za chwilę okaże się, że nie jest to konieczne.*

Drugą wadą powyższego rozwiązania jest to, że musimy wielokrotnie przeglądać te same elementy ciągu, a co za tym idzie najlepiej byłoby cały ciąg wczytać do tablicy. To w przypadku największego pliku wymaga tablicy o 1 000 000 elementów. W przypadku języka programowania C++ i 32-bitowego typu `int` wymaga to 4 000 000 bajtów, czyli około 4 gigabajtów pamięci. Nie jest to może dużo dla współczesnych komputerów, ale co gdyby dane liczyły nie milion, a miliard, bilion elementów?

Trzecią wadą, chyba najistotniejszą, jest powolność zaproponowanego algorytmu. Zastanówmy się, ile łącznie dodawań wykonamy w kroku (). Nietrudno zauważyć, że tych dodawań jest tyle, ile wynosi łączna suma długości wszystkich przedziałów. Tę wielkość łatwo oszacować z dołu. Przedziałów o długości co najmniej $n/3$ jest co najmniej $(n/3)^2$. A zatem łączna suma wszystkich przedziałów wynosi co najmniej $(n/3)^3$. To dla $n = 1\,000\,000$ daje więcej niż $3 \cdot 10^{16}$. Komputer wykonujący 10^9 dodawań na sekundę spędzałby na rozwiązywaniu naszego zadania więcej niż $3 \cdot 10^7$ sekund, a to jest więcej niż 8 tysięcy godzin. To jest trochę za długo jak na czas przeznaczony na rozwiązywanie zadań maturalnych!*

Jeden ze sposobów poszukiwania lepszych, szybszych algorytmów polega na przyjrzeniu się rozwiązaniu, które już mamy w ręku i zastanowieniu się, czy nie prowadzi ono do wykonywania wielu zbędnych operacji. Tak jest właśnie w tym przypadku. Zauważmy, że dla każdych dwóch bloków $b(i, j-1)$ i $b(i, j)$, różniących się tylko jednym elementem a_j , liczymy niezależnie dwie sumy $a_i + a_{i+1} + \dots + a_{j-1}$ oraz $a_i + a_{i+1} + \dots + a_j$, a przecież, żeby dostać drugą sumę wystarczy do pierwszej dodać tylko a_j . Jaki zysk! Zamiast $j-i+1$ dodawań wykonujemy tylko jedno. Ten pomysł pozwala nam natychmiast zaproponować następujący algorytm:

```
najdluzszy = 0; // długość najdłuższego z dotychczas
                //przejrzanych bloków trójkowych
for (i = 1; i <= n; i++){
    //liczymy sumy elementów w blokach o początkach na pozycji i
    //wartością suma będzie suma elementów ostatniego
    //przetworzonego bloku; inicjalnie suma == 0
    suma = 0;
    for (j = i; j <= n; j++){
        // obliczamy sumę elementów w bloku b(i,j)
        // suma jest równa sumie elementów w bloku b(i,j-1)
        // plus a_j
        suma = suma + a_j; // (*)
    }
}
```

```

// sprawdzamy, czy suma jest podzielna przez 3, czyli
// czy reszta z dzielenia suma przez 3 daje 0
if (suma % 3 == 0) {
    // jeśli jest podzielna przez 3 i blok b(i,j)
    // jest dłuższy od dotychczas najdłuższego bloku
    // trójkowego, to zapamiętujemy jego długość
    dl = j - i + 1;
    if (dl >= najdluzszy)
        najdluzszy = dl;
}
}

```

// najdluzszy jest długością najdłuższego bloku trójkowego w
// ciągu a

Ile tym razem wykonujemy dodawań (*)? Nietrudno zauważyć, że tyle, ile jest bloków. Żeby policzyć sumę elementów w bloku $b(i,j)$, wykonujemy tylko jedno dodawanie – do sumy elementów z bloku $b(i,j-1)$ dodajemy a_j . Ile jest wszystkich bloków? Bloków o początku na pozycji 1 jest n , bloków o początku na pozycji 2 jest $n-1$, bloków o początku na pozycji 3 jest $n-2$, itd. Tak więc bloków o początku na pozycji i jest $n-i+1$. Wszystkich bloków jest $n + n-1 + \dots + 1 = n(n-1)/2$. Dla $n = 1\,000\,000$ ta wartość wynosi $499\,999\,500\,000$. A zatem komputer wykonujący 10^9 dodawań na sekundę wykonałby nasze zadanie w około 500 sekund, czyli w około 7 minut. Natomiast dla ciągu o długości 30 000 odpowiedź dostalibyśmy natychmiast.

W przypadku rozpatrywanego zadania myślenie algorytmiczne może dać jeszcze lepsze efekty. Najpierw pozbadźmy się problemu dużych liczb. To łatwe. W wierszu (*) wystarczy sumować modulo 3. Inaczej mówiąc, w zmiennej suma zamiast sumy elementów pamiętamy resztę z dzielenia tej sumy przy dzieleniu przez 3. Więcej o wykonywaniu operacji arytmetycznych modulo napisaliśmy w komentarzu do zadania Podzielność. Przy tym podejściu wiersz (*) miałby postać:

```
suma = (suma + aj) % 3; // (*)
```

Natomiast instrukcja warunkowa `if (suma % 3 == 0)` przybrałaby postać:

```
if (suma == 0)
```

W ten sposób poradziłyśmy sobie z problemem dużych liczb, ale czasowa złożoność obliczeniowa naszego algorytmu pozostała bez zmian. Następujące spostrzeżenia pozwolą przyspieszyć poszukiwanie najdłuższego bloku trójkowego. Dla każdego $k = 1, 2, \dots, n$ oznaczmy przez s_k sumę pierwszych k elementów w ciągu a . Innymi słowy s_k jest sumą elementów w bloku $b(1,k)$. Dla wygody przyjmijmy, że mamy też element $a_0 = 0$ i w naturalny sposób weźmy $s_0 = 0$. Zauważmy teraz, że suma elementów w bloku $b(i,j)$, $1 \leq i \leq j \leq n$, jest równa $s_j - s_{i-1}$. Dla naszych celów wartości s wystarczy liczyć modulo 3. W takim przypadku wartością s_k może być tylko 0, 1 lub 2. Teraz najważniejsze:

Blok $b(i,j)$ jest blokiem trójkowym wtedy i tylko, gdy s_j oraz s_{i-1} mają taką samą wartość 0, 1, lub 2.

Powyższe spostrzeżenie daje bardzo proste rozwiązanie naszego zadania. Dla każdej wartości $w = 0, 1, 2$ poszukujemy pierwszej i ostatniej pozycji, dla której wartości sum s są takie same. Najbardziej odległe pozycje wyznaczają długość najdłuższego bloku trójkowego. Pozostaje jeszcze pytanie, czy zawsze taki blok istnieje. Osobom o zainteresowaniach bardziej matematycznych proponujemy wykazanie, że w każdym ciągu o długości co najmniej 3 taki blok musi istnieć. Dla naszych potrzeb przyjmijmy, że blok długości 0 jest blokiem trójkowym. Wówczas, jeśli wynikiem działania naszego algorytmu jest 0, oznacza to, że żaden blok w danym ciągu nie jest trójkowy.

Poniżej przedstawiamy program napisany w języku C++, który konkretyzuje opisane powyżej idee.

```
#include <iostream>
#include <algorithm>
using namespace std;

int najdluzszy_blok() {
    int poczatki = {0, -1, -1};
    //poczatki[w] - pierwsza pozycja, dla której suma s jest
    //równa w
    // -1 oznacza, że takiej pozycji jeszcze nie
    // znaleziono
    int suma_mod_3 = 0;
    int liczba_wczytanych = 0;
    int element;
    int najdluzszy;

    while (cin >> element) {
        liczba_wczytanych++;
        suma_mod_3 = (suma_mod_3 + element) % 3; //(*)
        if (poczatki[suma_mod_3] == -1)
            poczatki[suma_mod_3] = liczba_wczytanych;
        else
            najdluzszy = max(najdluzszy, liczba_wczytanych
            - poczatki[suma_mod_3])
    }

    return najdluzszy;
}

int main() {

    cout << "Dł. bloku: " << najdluzszy_blok() << endl;
    return 0;
}
```

Na koniec zauważmy, że w tym algorytmie liczba dodawań (*) wynosi tylko n i na dodatek nie musieliśmy najpierw wczytać całego ciągu do tablicy.

Opinia Konferencji Rektorów Akademickich Szkół Polskich o informatorach maturalnych od 2015 roku

Konferencja Rektorów Akademickich Szkół Polskich z wielką satysfakcją odnotowuje konsekwentne dążenie systemu oświaty do poprawy jakości wykształcenia absolwentów szkół średnich. Konferencja z uwagą obserwuje kolejne działania Ministerstwa Edukacji Narodowej w tym zakresie, zdając sobie sprawę, że od skuteczności tych działań w dużym stopniu zależą także efekty kształcenia osiągane w systemie szkolnictwa wyższego. W szczególności dotyczy to kwestii właściwego przygotowania młodzieży do studiów realizowanych z uwzględnieniem nowych form prowadzenia procesu kształcenia.

Podobnie jak w przeszłości, Konferencja konsekwentnie wspiera wszystkie działania zmierzające do tego, by na uczelnie trafiali coraz lepiej przygotowani kandydaci na studia. Temu celowi służyła w szczególności pozytywna opinia Komisji Edukacji KRASP z 2008 roku w sprawie nowej podstawy programowej oraz uchwała Zgromadzenia Plenarnego KRASP z dn. 6 maja 2011 r. w sprawie nowych zasad egzaminu maturalnego.

Z satysfakcją dostrzegamy, że ważne zmiany w egzaminie maturalnym, postulowane w cytowanej wyżej uchwale zostały praktycznie wdrożone przez MEN poprzez zmianę odpowiednich rozporządzeń.

Przedłożone do zaopiniowania informatory o egzaminach maturalnych opisują formę poszczególnych egzaminów maturalnych, przeprowadzanych na podstawie wymagań określonych w nowej podstawie programowej, a także ilustrują te wymagania wieloma przykładowymi zadaniami egzaminacyjnymi.

Po zapoznaniu się z przedłożonymi materiałami, KRASP z satysfakcją odnotowuje:

w zakresie języka polskiego:

- wzmocnienie roli umiejętności komunikacyjnych poprzez odejście od prezentacji na egzaminie ustnym i zastąpienie jej egzaminem ustnym, na którym zdający będzie musiał ad hoc przygotować samodzielną wypowiedź argumentacyjną,
- rezygnację z klucza w ocenianiu wypowiedzi pisemnych,
- zwiększenie roli tekstów teoretycznoliterackich i historycznoliterackich na maturze rozszerzonej;

w zakresie historii:

- kompleksowe sprawdzanie umiejętności z zakresu chronologii historycznej, analizy i interpretacji historycznej oraz tworzenia narracji historycznej za pomocą rozbudowanej wypowiedzi pisemnej na jeden z zaproponowanych tematów, łącznie pokrywających wszystkie epoki oraz obszary historii;

w zakresie wiedzy o społeczeństwie:

- położenie silniejszego akcentu na sprawdzanie umiejętności złożonych (interpretowanie informacji, dostrzeganie związków przyczynowo-skutkowych) w oparciu o poszerzony zasób materiałów źródłowych: teksty (prawne, naukowe, publicystyczne), materiały statystyczne, mapy, rysunki itp.

w zakresie matematyki:

- istotne zwiększenie wymagań na poziomie rozszerzonym poprzez włączenie zadań z rachunku różniczkowego i pojęć zaawansowanej matematyki,
- istotne poszerzenie wymagań z zakresu kombinatoryki oraz teorii prawdopodobieństwa;

w zakresie biologii oraz chemii:

- zwiększenie znaczenia umiejętności wyjaśniania procesów i zjawisk biologicznych i chemicznych,
- mierzenie umiejętności analizy eksperymentu – sposobu jego planowania, przeprowadzania, stawianych hipotez i wniosków formułowanych na podstawie dołączonych wyników;

w zakresie fizyki:

- zwiększenie znaczenia rozumienia istoty zjawisk oraz tworzenie formuł matematycznych łączących kilka zjawisk,
- mierzenie umiejętności planowania i opisu wykonania prostych doświadczeń, a także umiejętności analizy wyników wraz z uwzględnieniem niepewności pomiarowych;

w zakresie geografii:

- uwzględnienie interdyscyplinarności tej nauki poprzez sprawdzanie umiejętności integrowania wiedzy z nauk przyrodniczych do analizy zjawisk i procesów zachodzących w środowisku geograficznym,
- znaczne wzbogacenie zasobu materiałów źródłowych (mapy, wykresy, tabele statystyczne, teksty źródłowe, barwne zdjęcia, w tym lotnicze i satelitarne), także w postaci barwnej.

Konferencja Rektorów Akademickich Szkół Polskich z zadowoleniem przyjmuje też informację o wprowadzeniu na świadectwach maturalnych od 2015 roku dodatkowej formy przedstawiania wyniku uzyskanego przez zdającego w postaci jego pozycji na skali centylowej, tj. określenie, jaki odsetek zdających uzyskał taki sam lub słabszy wynik od posiadacza świadectwa. Wprowadzenie tej dodatkowej skali uwolni szkoły wyższe od dotychczasowego dylematu odnoszenia do siebie surowych wyników kandydatów na studia rekrutowanych na podstawie wyników egzaminów maturalnych o istotnie różnym poziomie trudności – rekrutacja stanie się prostsza i bardziej obiektywna.

Reasumując, w opinii Konferencji Rektorów Akademickich Szkół Polskich zaprezentowana w przedłożonych informatorach forma matury istotnie przyczyni się do tego, że młodzież przekraczająca progi uczelni będzie lepiej przygotowana do podjęcia studiów wyższych.

5 lipca 2013 r.

Przewodniczący KRASP

prof. zw. dr hab. Wiesław Banyś