

(في الامتحان فصلان، ولكل سؤال 25 علامة. في الفصل الأول يجب الطالب عن سؤالين من الأسئلة 1 حتى 3. يكتب الطالب البرامج بلغة واحدة فقط: Java أو C#. هنا نكتب الحل باللغتين: Java أولاً ثم C#.)

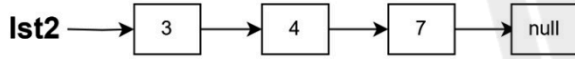
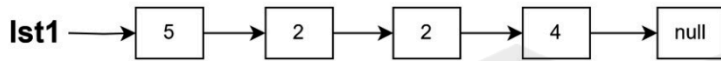
الفصل الأول (50 درجة)

أجبوا عن اثنين من الأسئلة 1-3. (لكل سؤال – 25 درجة.)

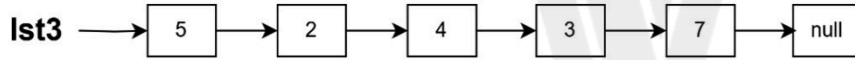
1. معطاة قائمتان لأعداد صحيحة Ist1 و Ist2.

أ. اكتبوا عملية خارجية باسم createUnion بلغة Java أو CreateUnion بلغة C#.

تتلقى العملية القائمتين كمدخل، وتعيد قائمة جديدة تحوي جميع الأعداد التي تظهر في إحدى القائمتين على الأقل. ترتيب الأعداد في القائمة المُعادة ليس مهماً، وكل عدد يظهر مرة واحدة فقط. مثال: بالنسبة لاستدعاء العملية مع القائمتين Ist1, Ist2 اللتين أمامكم:



تُعاد القائمة التالية (لا توجد أهمية لترتيب المتغيرات):

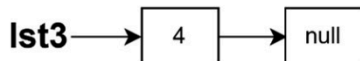


الشرح: كل عدد ظهر على الأقل في إحدى القائمتين، يظهر في القائمة الجديدة. وكذلك في القائمة الجديدة كل عدد يظهر مرة واحدة فقط (بدون مضاعفات).

ب. اكتبوا عملية خارجية باسم intersection بلغة Java أو Intersection بلغة C#.

تتلقى العملية كمدخل القائمتين وتعيد قائمة جديدة تحوي فقط الأعداد التي تظهر في Ist1 وكذلك في Ist2. ترتيب الأعداد في القائمة المُعادة ليس مهماً، وكل عدد يظهر مرة واحدة فقط. إذا لم يوجد أي عدد مشترك، تُعيد العملية قائمة فارغة.

مثال: بالنسبة لاستدعاء العملية مع القائمتين Ist1, Ist2 المعطيتين في المثال الذي في البند "أ"، تُعاد القائمة التالية:



الشرح: العدد 4 فقط، يظهر في القائمتين.

الحل: أ. نمرّ على القائمتين ونضيف كل عدد غير موجود بعد

ب. نضيف عدداً من Ist1 موجوداً في Ist2 وغير مكرّر

(الحلّ والشرح: فيفو · vevo. لا تنشر وزارة التربية حلولًا لهذا الامتحان. لا يوجد مترجم Java ولا C# على جهازنا، لذلك ترجمنا الكود سطرًا سطرًا إلى Python وشغلناه فعليًا على مثال الورقة وعلى حالات حديثة وعلى خمسة آلاف زوج قوائم عشوائية، وقارنًا بالاتّحاد والتقاطع الصحيحين).

الحلّ الكامل:

عملية مساعدة للبدين

نحتاج في البدين إلى سؤال واحد كثيرًا: هل العدد x موجود في قائمة؟ لذلك نكتب أولًا عملية مساعدة `isIn` تمرّ على القائمة من رأسها حتّى النهاية. السطر 3 هو قلب العملية: إذا وجدنا x نعيد `true` فورًا.

Java

```
1 public static boolean isIn(Node<Integer> lst, int x) {
2     while (lst != null) {
3         if (lst.getValue() == x)
4             return true;           // x was found
5         lst = lst.getNext();
6     }
7     return false;
8 }
```

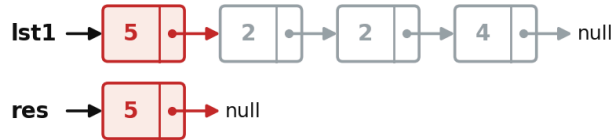
C#

```
1 public static bool IsIn(Node<int> lst, int x)
2 {
3     while (lst != null)
4     {
5         if (lst.GetValue() == x)
6             return true;           // x was found
7         lst = lst.GetNext();
8     }
9     return false;
10 }
```

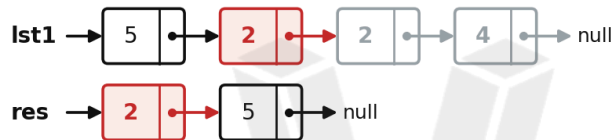
(المتغيّر `lst` داخل العملية نسخة من المؤشّر فقط. تحريكه لا يغيّر القائمة الأصلية، لذلك تبقى القائمتان كما هما.)

الفكرة: نبني قائمة جديدة res فارغة في البداية. نمرّ على حلقات lst1 ثمّ على حلقات lst2. كلّ عدد غير موجود في res نضيفه في رأسها. العدد الموجود فيها لا نضيفه مرّة ثانية، فلا تظهر مضاعفات. الترتيب غير مهمّ حسب الورقة، والإضافة في الرأس هي الأسهل.

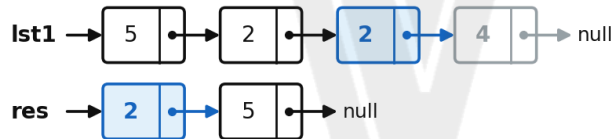
الخطوة 1: العدد 5 من lst1 غير موجود في res، فنضيفه في الرأس (بالأحمر).



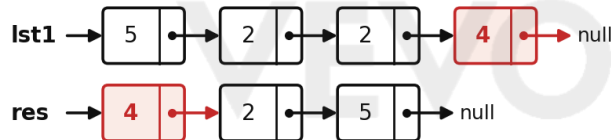
الخطوة 2: العدد 2 من lst1 غير موجود في res، فنضيفه في الرأس (بالأحمر).



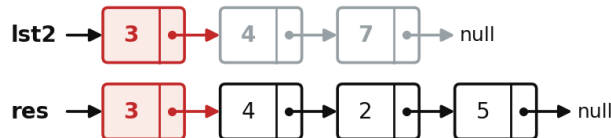
الخطوة 3: العدد 2 من lst1 موجود في res (بالأزرق)، فلا نضيفه.



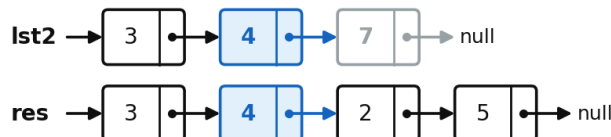
الخطوة 4: العدد 4 من lst1 غير موجود في res، فنضيفه في الرأس (بالأحمر).



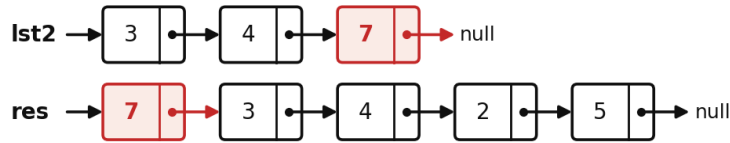
الخطوة 5: العدد 3 من lst2 غير موجود في res، فنضيفه في الرأس (بالأحمر).



الخطوة 6: العدد 4 من lst2 موجود في res (بالأزرق)، فلا نضيفه.



الخطوة 7: العدد 7 من lst2 غير موجود في res، فنضيفه في الرأس (بالأحمر).



(النتيجة 5 → 2 → 4 → 3 → 7 فيها الأعداد نفسها التي في مثال الورقة، كل عدد مرّة واحدة، الترتيب مختلف، والورقة تقول إن الترتيب ليس مهمًا.)

الكود بلغة Java. السطر 6 يفحص أن العدد غير موجود، والسطر 7 يضيفه في الرأس. السطر 14 والسطر 15 يمرّان على القائمتين بالعمليّة نفسها:

Java

```
1 // adds to res every value of src that is not in res yet
2 public static Node<Integer> addNew(Node<Integer> src,
3                                     Node<Integer> res) {
4     while (src != null) {
5         int x = src.getValue();
6         if (!isIn(res, x))
7             res = new Node<Integer>(x, res); // add at front
8         src = src.getNext();
9     }
10    return res;
11 }
12 public static Node<Integer> createUnion(Node<Integer> lst1,
13                                         Node<Integer> lst2) {
14     Node<Integer> res = addNew(lst1, null);
15     res = addNew(lst2, res);
16     return res;
17 }
```

الكود بلغة C# (المنطق نفسه؛ السطر 7 والسطر 8 يضيفان العدد الجديد، والسطر 15 والسطر 16 يمرّان على القائمتين):

C#

```
1 // adds to res every value of src that is not in res yet
2 public static Node<int> AddNew(Node<int> src, Node<int> res)
3 {
4     while (src != null)
5     {
6         int x = src.GetValue();
7         if (!IsIn(res, x))
8             res = new Node<int>(x, res); // add at front
9         src = src.GetNext();
10    }
11    return res;
12 }
13 public static Node<int> CreateUnion(Node<int> lst1, Node<int> lst2)
14 {
15     Node<int> res = AddNew(lst1, null);
16     res = AddNew(lst2, res);
17     return res;
18 }
```

(حالات حدّية شغلناها: قائمتان فارغتان تعطيان قائمة فارغة. قائمة أولى فارغة والثانية $1 \rightarrow 1 \rightarrow 2$ تعطيان قائمة فيها العدد 1 مرّة واحدة والعدد 2 مرّة واحدة. بعد التنفيذ تبقى القائمتان الأصليتان كما كانتا.)

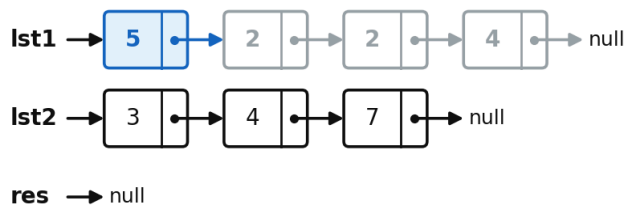
```
if (!IsIn(res, x)) res = new Node<Integer>(x, res);
```

الجواب النهائي للبند أ (قلب الحل)

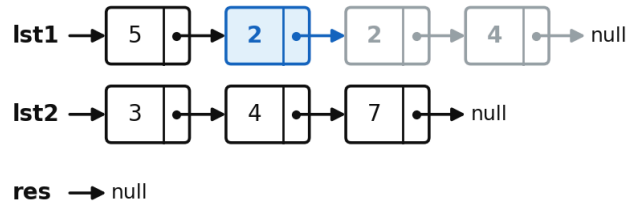
البند ب

الفكرة: نمرّ على حلقات lst1 فقط. نضيف العدد x إلى res بشرطين معًا: x موجود في lst2، ولكّنه غير موجود بعد في res. إذا لم يوجد أيّ عدد مشترك يبقى res مساويًا لـ null، وهذه هي القائمة الفارغة المطلوبة.

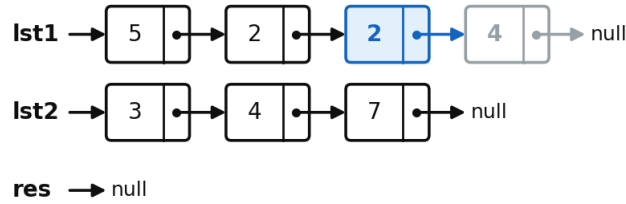
الخطوة 1: العدد 5 غير موجود في lst2 (بالأزرق)، فلا نضيفه.



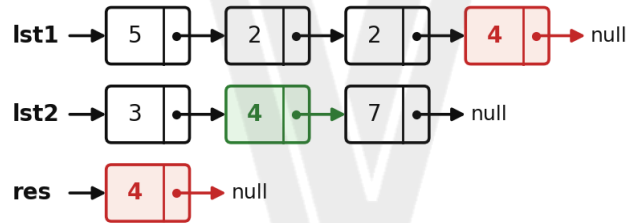
الخطوة 2: العدد 2 غير موجود في lst2، فلا نضيفه.



الخطوة 3: العدد 2 غير موجود في lst2، فلا نضيفه.



الخطوة 4: العدد 4 موجود في lst2 (بالأخضر) وغير موجود في res، فنضيفه (بالأحمر).



الكود بلغة Java. السطر 7 هو الشرط المزدوج، والسطر 8 يضيف العدد في الرأس:

Java

```
1 public static Node<Integer> intersection(Node<Integer> lst1,
2                                           Node<Integer> lst2) {
3     Node<Integer> res = null;
4     Node<Integer> pos = lst1;
5     while (pos != null) {
6         int x = pos.getValue();
7         if (isIn(lst2, x) && !isIn(res, x))
8             res = new Node<Integer>(x, res); // add at front
9         pos = pos.getNext();
10    }
11    return res; // null if there is no common number
12 }
```

C#

```
1 public static Node<int> Intersection(Node<int> lst1, Node<int> lst2)
2 {
3     Node<int> res = null;
4     Node<int> pos = lst1;
5     while (pos != null)
6     {
7         int x = pos.GetValue();
8         if (IsIn(lst2, x) && !IsIn(res, x))
9             res = new Node<int>(x, res);          // add at front
10        pos = pos.GetNext();
11    }
12    return res;          // null if there is no common number
13 }
```

(شغلنا الترجمة على مثال الورقة فأعادت قائمة فيها العدد 4 فقط. للقائمة 2 → 1 مع القائمة 3 أعادت null، أي قائمة فارغة.)

if (isIn(lst2, x) && !isIn(res, x)) res = new Node<Integer>(x, res);

الجواب النهائي للبند ب (للمثال تُعاد القائمة 4)

نص الأسئلة والصور: وزارة التربية والتعليم، النموذج 899271، صيف 2025. الحل والشرح: فيفو · vevo (لا تنشر وزارة التربية حلولاً لهذا الامتحان). حقوق الطبع للوزارة: «النسخ أو النشر ممنوعان إلا بإذن من وزارة التربية والتعليم».

VEVO