מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ד, 2024
מספר השאלון: 899381
תרגום לערבית (2)

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: صيف 2024
رقم النّموذج: 899381
ترجمة إلى العربيّة (2)

انتبهوا: في هذا الامتحان توجد توجيهات خاصّة.
يجب الإجابة عن الأسئلة حسب التّوجيهات.

מדעי המחשב

הוראות

א. משך הבחינה: שלוש שעות וחצי.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שלושה פרקים.
פרק ראשון — (10×1) + (15×1) — 25 נק'
פרק שני — (25×3) — 75 נק'
פרק שלישי —
סך הכול — 100 נק'

שימו לב: אם תבחרו לענות על שאלות מן הפרק השלישי, בחרו בשאלות מתוך מסלול אחד בלבד.

ג. חומר עזר מותר בשימוש: כל חומר עזר, חוץ ממחשבון שיש בו אפשרות תכנות.

ד. הוראות מיוחדות:
1. אם תבחרו לענות על שאלות **רק** מן הפרק הראשון והפרק השני, **רשמו על הכריכה החיצונית** של המחברת **ללא מסלול**, אחרת רשמו את **שם המסלול שלמדתם**.
המסלול הוא אחד משלושת המסלולים האלה:
אלגוריתמים, מודלים חישוביים,
תכנות מונחה עצמים.
2. את כל התוכניות שיש לכתוב בשפת מחשב בפרקים הראשון והשני כתבו בשפה אחת בלבד — Java או C#.

הערה: לא יוּרדוּ נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

علم الحاسوب

تعليمات

أ. مدّة الامتحان: ثلاث ساعات ونصف.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج ثلاثة فصول.
الفصل الأوّل — (10×1) + (15×1) — 25 درجة
الفصل الثّاني — (25×3) — 75 درجة
الفصل الثّالث —
المجموع — 100 درجة

انتبهوا: إذا اخترتم الإجابة عن أسئلة من الفصل الثّالث، اختاروا أسئلة من مسار واحد فقط.

جـ. موادّ مساعدة يُسمح استعمالها: كلّ مادّة مساعدة، عدا الحاسبة التي توجد فيها إمكانيّة برمجة.

د. تعليمات خاصّة:
1. إذا اخترتم الإجابة عن أسئلة من الفصل الأوّل والفصل الثّاني **فقط**، **اكتبوا على الغلاف الخارجيّ** للدّفتر "**بدون مسار**"، خلاف ذلك اكتبوا **اسم المسار الّذي تعلّمتموه**.
المسار هو أحد المسارات الثّلاثة التّالية:
ألغوريثمات، موديلات حسابيّة،
برمجة موجّهة كائنات.
2. اكتبوا جميع البرامج التي يجب كتابتها بلغة حاسوب في الفصلين الأوّل والثّاني، بلغة واحدة فقط — Java أو C#.

ملاحظة: لن تُخصم درجات إذا كتبتم في البرامج حرفًا كبيرًا بدلًا من حرف صغير أو بالعكس.

يجب الكتابة في دفتر الامتحان فقط. يجب كتابة "مسوّدة" في بداية كلّ صفحة تُستعمَل مسوّدة.
كتابة أيّة مسوّدة على أوراق خارج دفتر الامتحان قد تسبّب إلغاء الامتحان.

الأسئلة في هذا النّموذج ترد بصيغة الجمع، ورغم ذلك يجب على كلّ طالبة وطالب الإجابة عنها بشكل فرديّ.

ב ה צ ל ח ה !

نتمنّى لكم النّجاح!

الأسئلة

ملاحظة: في كلّ سؤال يُطلب فيه استقبال، لا حاجة لفحص سلامة المدخَلات.
للّذين يحلّون بلغة Java: في كلّ سؤال يُطلب فيه استقبال، افترِضوا أنّ الأمر التالي مكتوب في البرنامج:

```
Scanner input = new Scanner (System.in);
```

الفصل الأوّل (25 درجة)

أجيبوا عن السؤال 1 – إلزاميّ (10 درجات).

1. اكتبوا عمليّة خارجيّة باسم biggestSum بلغة Java أو BiggestSum بلغة #C، تتلقّى مصفوفة من نمط صحيح – arr، ليس فيها أعداد سالبة (الأعداد أكبر أو تساوي 0). العدد 0 يظهر في المصفوفة مرّتَيْن على الأقلّ.
تُعيد العمليّة أكبر مجموع للأعداد، الذي يَنْتُج بين كلّ عددَي 0 في المصفوفة.
افترِضوا أنّه توجد أعداد موجبة بين كلّ عددَي 0 في المصفوفة.
مثال:
بالنسبة للمصفوفة التي أمامكم، تُعيد العمليّة 17.

	0	1	2	3	4	5	6	7	8	9	10	11
arr	33	0	**5**	**4**	0	**17**	0	**4**	**10**	0	5	14
			9			17		14				

الشرح: 17 هو أكبر مجموع من بين مجاميع الأعداد، التي تَنْتُج بين كلّ عددَي 0 في المصفوفة (14, 17, 9).

/ يتبع في صفحة 3 /

أجيبوا عن أحد السؤالين 2-3 (15 درجة) .

2. بعد الانتخابات، بدأ سكرتير الكنيست بتخصيص أماكن لأعضاء الكنيست في لجان الكنيست المختلفة .
لهذا الغرض، عُرِّفت الفئتان التاليتان : **Member** – عضو كنيست، وَ **Committee** – لجنة .
للفئة Member صفتان :

- name – اسم عضو الكنيست (افترِضوا أنّه لا يوجد عضوا كنيست اسماهما متشابهان) .
- isCoal – قيمة بوليانيّة تتلقّى true إذا كان عضو الكنيست يتبع للائتلاف الحكوميّ، خلاف ذلك تتلقّى false .

للفئة Committee ثلاث صفات :

- name – اسم اللجنة .
- members – مصفوفة من نمط Member تحوي أعضاء الكنيست الذين تمّ تخصيص أماكن لهم في اللجنة .
- count – عدد أعضاء اللجنة الحاليّين في اللجنة (أعضاء اللجنة يُحفَظون بتسلسل من بداية المصفوفة) .

افترِضوا أنّه توجد عمليّتا get/Get وَ set/Set لكلّ واحدة من صفات الفئتين .

أ. اكتبوا عنوانَي الفئتين Member وَ Committee ، وصفاتهما .

ب. اكتبوا عمليّة داخليّة في الفئة Committee باسم total بلغة Java أو Total بلغة #C، تتلقّى بارامترًا بوليانيًّا – belong .
إذا كان belong هو true ، تُعيد العمليّة عدد أعضاء الكنيست في اللجنة الذين يتبعون للائتلاف الحكوميّ، خلاف ذلك (إذا كان belong هو false) تُعيد العمليّة عدد أعضاء الكنيست في اللجنة الذين لا يتبعون للائتلاف الحكوميّ .

عند تخصيص أماكن لأعضاء الكنيست في اللجان، يحرص سكرتير الكنيست على القاعدتَيْن التاليتَيْن :

- في كلّ لجنة، عدد أعضاء الكنيست الذين يتبعون للائتلاف الحكوميّ يكون أكبر من عدد أعضاء الكنيست الذين لا يتبعون للائتلاف الحكوميّ .
- في كلّ لجنة، يكون عدد أعضاء الكنيست أقلّ من 16 .

جـ. طبِّقوا العمليّة الخارجيّة التي أمامكم :

Java: public static int amount (Committee [] arr, Member m)

C#: public static int Amount (Committee [] arr, Member m)

تتلقّى العمليّة مصفوفة كاملة – arr لجميع لجان الكنيست، وعضو كنيست معيَّن – m ، الذي لم يتمّ بعد تخصيص مكان له في أيّة لجنة .
تُعيد العمليّة عدد اللجان التي من الممكن أن يتمّ تخصيص مكان لعضو الكنيست فيها (حسب قاعدتَي سكرتير الكنيست) .
يجب استعمال العمليّة التي كتبتموها في البند "ب" .

/ يتبع في صفحة 4 /

3. "مصفوفة آلات إنتاج البراغيّ" هي مصفوفة من نمط صحيح، يظهر في كلّ واحدة من خلاياها عدد **الثواني** الذي تحتاجه آلة واحدة معيَّنة لإنتاج برغيّ **واحد** (كلّما كان عدد الثواني أقلّ، أنتجت الآلة براغيّ بسرعة أكبر). المصفوفة ليست مرتَّبة.

مثال: في المصفوفة التي أمامكم، تحتاج الآلة التي في المؤشّر 0 إلى ثانية واحدة فقط لإنتاج برغيّ واحد، وتحتاج الآلة التي في المؤشّر 1 إلى 9 ثوانٍ لإنتاج برغيّ واحد، وتحتاج الآلة التي في المؤشّر 2 إلى 3 ثوانٍ لإنتاج برغيّ واحد.

"مصفوفة آلات إنتاج البراغيّ"

0	1	2
1	9	3

في مصنع لإنتاج البراغيّ "البراغيّ"، توجد عدّة آلات لإنتاج البراغيّ. يَحْفَظ المصنع في "مصفوفة آلات إنتاج البراغيّ" – arr عدد الثواني الذي تحتاجه كلّ واحدة من آلاته لإنتاج برغيّ واحد. يَستعمل المصنع جميع الآلات لإنتاج البراغيّ في نفس الوقت.

أ. اكتبوا عمليّة خارجيّة باسم total بلغة Java أو Total بلغة C#، تتلقّى المصفوفة arr، وعددًا صحيحًا من الثواني – numSeconds.
تُعيد العمليّة العدد الكلّيّ للبراغيّ التي يُنْتِجها المصنع بواسطة جميع آلاته خلال numSeconds.

مثال: بالنسبة للمصفوفة التي في المثال أعلاه وَ numSeconds = 5، تُعيد العمليّة 6.
الشرح: كمّيّة الإنتاج للآلات في فترة زمنيّة مقدارها 5 ثوانٍ مفصَّلة أمامكم:
الآلة في المؤشّر 0 تُنْتِج خمسة براغيّ (برغيّ واحد في كلّ ثانية)،
والآلة في المؤشّر 1 لا تُنْتِج أيّ برغيّ (لأنّها تحتاج إلى 9 ثوانٍ لإنتاج برغيّ واحد)،
والآلة في المؤشّر 2 تُنْتِج برغيًّا واحدًا.
المجموع الكلّيّ: ستّة براغيّ (1 + 5).

ب. اكتبوا عمليّة خارجيّة باسم minTime بلغة Java أو MinTime بلغة C#، تتلقّى المصفوفة arr، وعددًا صحيحًا من البراغيّ – amount.
تُعيد العمليّة أصغر عدد ممكن من الثواني التي يحتاجها المصنع كي يستطيع إنتاج على الأقلّ ما مجموعه amount براغيّ (من جميع الآلات معًا).
يمكن استعمال العمليّة التي كتبتموها في البند "أ".

مثال: بالنسبة للمصفوفة التي في المثال أعلاه وَ amount = 7، تُعيد العمليّة 6.
الشرح: كمّيّة الإنتاج للآلات في فترة زمنيّة مقدارها 6 ثوانٍ مفصَّلة فيما يلي:
الآلة في المؤشّر 0 تُنْتِج ستّة براغيّ،
والآلة في المؤشّر 1 لا تُنْتِج أيّ برغيّ،
والآلة في المؤشّر 2 تُنْتِج برغيَّيْن.
المجموع الكلّيّ: ثمانية براغيّ (2 + 6).
أي، تحتاج الآلات الثلاث إلى 6 ثوانٍ لتُنْتِج، على الأقلّ ما مجموعه سبعة براغيّ. في أقلّ من 6 ثوانٍ، لن تتمكّن الآلات من إنتاج سبعة براغيّ (مثلًا في فترة زمنيّة تبلغ 5 ثوانٍ، تُنْتِج الآلات الثلاث معًا ما مجموعه ستّة براغيّ، كما هو موصوف في المثال في البند "أ").

/ يتبع في صفحة 5 /

يجب الإجابة عن ما مجموعه ثلاثة أسئلة من الفصلين الثّاني والثّالث (لكلّ سؤال – 25 درجة).

الفصل الثاني

انتبهوا: في كلّ سؤال يُطلب فيه تطبيق، بإمكانكم استعمال عمليّات الفئات:
دَوْر وراصّة وشجرة بينارية وحلقة، بدون تطبيقها. إذا استعملتم عمليّات إضافيّة، يجب تطبيقها.

4. "حدّ سحريّ" هو حدّ في دَوْر أعداد، قيمته تساوي مجموع قيمتَي الحدّ الذي قبله والحدّ الذي بعده.
ملاحظة: العدد الأوّل في الدَّوْر والعدد الأخير في الدَّوْر ليسا "حدَّيْن سحريَّيْن".

أ. اكتبوا عمليّة باسم isMagic بلغة Java أو IsMagic بلغة C#، تتلقّى دَوْرًا – q من نمط صحيح، وعددًا صحيحًا – m أكبر من 0 وأصغر أو مساوٍ لِكِبَر الدَّوْر.
تُعيد العمليّة true إذا كان الحدّ في المكان الـ m في الدَّوْر هو "حدّ سحريّ"، خلاف ذلك، تُعيد العمليّة false.

ملاحظتان:
- في نهاية العمليّة، يجب الحفاظ على مبنى الدَّوْر كما تمّ تلقّيه.
- **لا** تستعمِلوا في هذا البند مصفوفة أو قائمة مرتبطة. الحلّ الذي يتضمّن استعمالهما، لن يحصل على درجات.

مثال: بالنسبة للدَّوْر الذي أمامكم:

	رأس الدَّوْر						نهاية الدَّوْر	
	1	2	3	4	5	6	7	
	5	11	6	9	3	6	3	

بالنسبة لِـ $m = 1$، تُعيد العمليّة false (العدد الأوّل في الدَّوْر ليس "حدًّا سحريًّا").
بالنسبة لِـ $m = 2$، تُعيد العمليّة true $(11 = 6 + 5)$.
بالنسبة لِـ $m = 3$، تُعيد العمليّة false $(6 \neq 9 + 11)$.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

ب. اكتبوا عمليّة باسم nMagic بلغة Java أو NMagic بلغة C# ، تتلقّى دَوْرًا من نمط صحيح – q ، وعددًا صحيحًا n أكبر من 0 وأصغر أو مساوٍ لِكِبَر الدَّوْر.

تُعيد العمليّة true إذا كانت جميع الحدود الواقعة في الأماكن التي هي مضاعفات لِـ n (المكان الـ n في الدَّوْر والمكان الـ 2n في الدَّوْر وهكذا، بتخطّيات n أماكن) هي "حدود سحريّة". خلاف ذلك، تُعيد العمليّة false.

يمكن استعمال العمليّة التي كتبتموها في البند "أ".

ملاحظتان:
- في هذه العمليّة، لا حاجة للحفاظ على الدَّوْر الذي تمّ تلقّيه.
- **لا** تستعمِلوا في هذا البند مصفوفة أو قائمة مرتبطة. الحلّ الذي يتضمّن استعمالهما، لن يحصل على درجات.

أمثلة: بالنسبة للدَّوْر الذي في صفحة 5:

بالنسبة لِـ $n = 2$ ، تُعيد العمليّة true ، لأنّ جميع الحدود الواقعة في الأماكن التي هي مضاعفات لِـ 2 (6 ، 4 ، 2) هي "حدود سحريّة".

بالنسبة لِـ $n = 4$ ، تُعيد العمليّة true ، لأنّ الحدّ الواقع في المكان 4 هو "حدّ سحريّ" (لا توجد في الدَّوْر حدود أخرى في الأماكن التي هي مضاعفات لِـ 4).

بالنسبة لِـ $n = 3$ ، تُعيد العمليّة false ، لأنّ الحدّ في المكان 3 ليس "حدًّا سحريًّا".

/ يتبع في صفحة 7 /

5. معطاة الفئة **Patient** – مريض في غرفة الطوارئ، ولها صفتان:

- id – رقم هويّة المريض، من نمط صحيح.
- priority – مستوى أولويّة علاج المريض. مستوى الأولويّة ممثَّل بواسطة عدد من نمط صحيح بين 1 إلى 10 .
  كلّما كان العدد أعلى، كان مستوى الأولويّة أعلى.

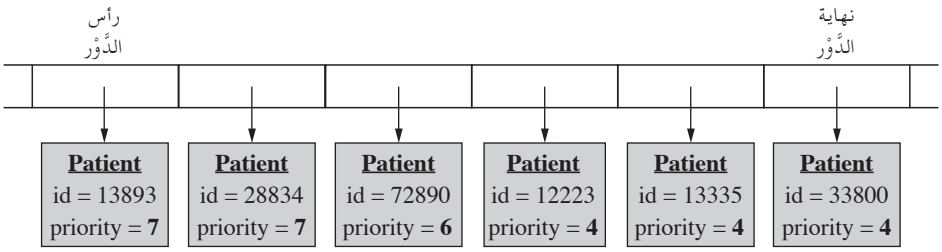
افترِضوا أنّ لصفتَي الفئة توجد عمليّتا get/Get وَ set/Set .

ترتيب علاج المرضى في غرفة الطوارئ يتمّ على النحو التالي:
كلّما كان مستوى أولويّة العلاج أعلى، أُعطيت أسبقيّة لعلاج المريض. إذا كان هناك أكثر من مريض واحد في نفس مستوى الأولويّة، تُعطى أسبقيّة لعلاج المريض الذي وصل قبل غيره إلى غرفة الطوارئ.

من أجل الحفاظ على ترتيب العلاج، بُنيت الفئة **PriorQueue** – دَوْر أفضليّات، ولها صفة واحدة:

- q – توجيه إلى دَوْر، من نمط Patient .

مثال لدَوْر q:



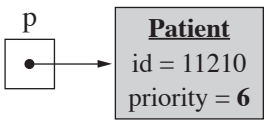
افترِضوا أنّ المرضى الذين مستوى أولويّتهم متطابق معروضون في المثال حسب ترتيب وصولهم إلى غرفة الطوارئ.

أ. طبِّقوا العمليّة التي أمامكم التي تتبع لواجهة تطبيق الفئة PriorQueue :

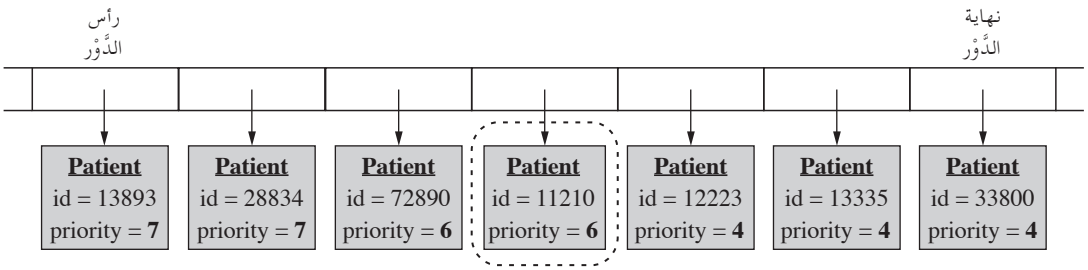
Java – public void priorityInsert (Patient p)

C# – public void PriorityInsert (Patient p)

تتلقّى العمليّة مريضًا جديدًا – p ، وتدمجه في الدَّوْر q بموجب قواعد غرفة الطوارئ المكتوبة أعلاه.
مثال: بالنسبة للدَّوْر المعروض أعلاه والكائن الذي أمامكم:



يبدو الدَّوْر هكذا بعد الدَّمْج:



(انتبهوا: تكملة السؤال في الصفحة التالية.)

ب. في بعض الأحيان، يتغيّر مستوى أولويّة مريض معيَّن أثناء وجوده في غرفة الطوارئ.
طبِّقوا العمليّة التي أمامكم التي تتبع إلى واجهة تطبيق الفئة PriorQueue :

Java – public void update (int id, int pri)

C# – public void Update (int id, int pri)

تتلقّى العمليّة رقم هويّة مريض موجود في الدَّوْر – id ، وعددًا – pri يمثّل مستوى أولويّته المُحدَّث.
تُحدِّث العمليّة الصفة priority للمريض وتدمجه في دَوْر العلاج في المكان المناسب له (حسب مستوى الأولويّة المُحدَّث – pri) . إذا سبق وكان في الدَّوْر مرضى آخرون بنفس مستوى الأولويّة – pri ، فإنّ المريض الحاليّ – id يُدمَج بعدهم وكأنّه وصل بعدهم إلى غرفة الطوارئ.

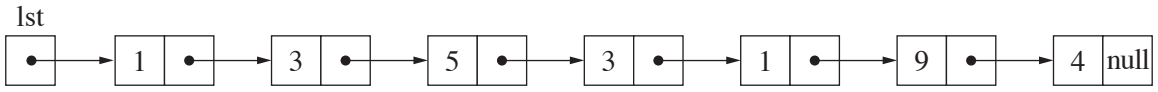
/ يتبع في صفحة 9 /

6. انتبهوا: لهذا السؤال صيغتان: بلغة Java في صفحة 9 وبلغة C# في صفحة 10.
للّذين يحلُّون بلغة Java

أ. معطاة العمليّة what :

```
public static Node<Integer> what (Node<Integer>lst, int x)
{
    if (lst == null)
        return null;
    Node<Integer> temp = what (lst.getNext(),x);
    if (lst.getValue() == x)
        return temp;
    lst.setNext (temp);
    return lst;
}
```

معطاة سلسلة حلقات – lst ، من نمط صحيح:

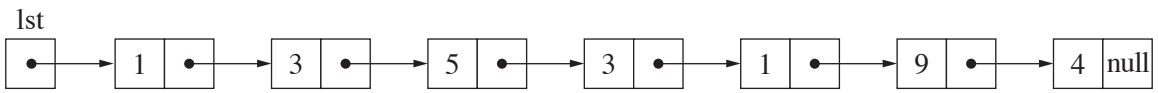


(1) تتبَّعوا العمليّة what (lst, 1) ، واعرِضوا السلسلة التي تُعيدها العمليّة.
(2) ماذا تفعل العمليّة what ؟ اشرحوا إجابتكم.
(3) ما هي تعقيدات العمليّة what ؟ علِّلوا إجابتكم.

ب. معطاة العمليّة guess :

```
public static void guess (Node<Integer>lst)
{
    if (lst != null)  {
        Node<Integer> temp = what (lst.getNext(), lst.getValue());
        lst.setNext (temp);
        guess (lst.getNext());
    }
}
```

معطاة سلسلة حلقات – lst ، من نمط صحيح:



(1) تتبَّعوا العمليّة guess (lst) ، واعرِضوا السلسلة lst في نهاية العمليّة.
في هذا البند، لا حاجة لتتبُّع العمليّة what .
(2) ماذا تفعل العمليّة guess ؟ اشرحوا إجابتكم.
(3) ما هي تعقيدات العمليّة guess ؟ علِّلوا إجابتكم.

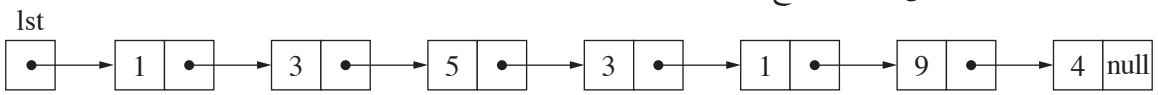
/ يتبع في صفحة 10 /

للّذين يحلُّون بلغة C#

أ. معطاة العمليّة What :

```
public static Node<int> What (Node<int>lst, int x)
{
    if (lst == null)
        return null;
    Node<int> temp = What (lst.GetNext(),x);
    if (lst.GetValue() == x)
        return temp;
    lst.SetNext (temp);
    return lst;
}
```

معطاة سلسلة حلقات – lst ، من نمط صحيح :



(1) تتبَّعوا العمليّة What (lst, 1) ، واعرِضوا السلسلة التي تُعيدها العمليّة.

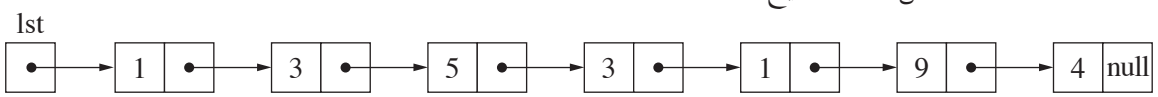
(2) ماذا تفعل العمليّة What ؟ اشرحوا إجابتكم.

(3) ما هي تعقيدات العمليّة What ؟ علِّلوا إجابتكم.

ب. معطاة العمليّة Guess :

```
public static void Guess (Node<int>lst)
{
    if (lst != null)  {
        Node<int> temp = What (lst.GetNext(), lst.GetValue());
        lst.SetNext (temp);
        Guess (lst.GetNext());
    }
}
```

معطاة سلسلة حلقات – lst ، من نمط صحيح :



(1) تتبَّعوا العمليّة Guess (lst) ، واعرِضوا السلسلة lst في نهاية العمليّة.
في هذا البند، لا حاجة لتتبُّع العمليّة What .

(2) ماذا تفعل العمليّة Guess ؟ اشرحوا إجابتكم.

(3) ما هي تعقيدات العمليّة Guess ؟ علِّلوا إجابتكم.

/ يتبع في صفحة 11 /

7. معطاة الفئة **BusStation** – محطّة باص، ولها ثلاث صفات:

- num – رقم المحطّة، من نمط صحيح.
- arr – مصفوفة من نمط صحيح بكِبَر 10 ، تحوي أرقام خطوط الباصات التي تتوقّف في المحطّة. يتوقّف في المحطّة حتّى 10 خطوط، وتُحفَظ هذه الخطوط بتسلسل من بداية المصفوفة.
- amount – عدد خطوط الباصات التي تتوقّف في المحطّة بالفعل، من نمط صحيح. يتوقّف في المحطّة خطّ باص واحد على الأقلّ.

افترِضوا أنّ لصفات الفئة توجد عمليّتا get/Get وَ set/Set .

أ. طبِّقوا العمليّة التي أمامكم التي تتبع لواجهة تطبيق الفئة BusStation :

Java – public boolean isStopping (int n)

C# – public bool IsStopping (int n)

تتلقّى العمليّة رقم خطّ باص – n من نمط صحيح. تُعيد العمليّة true إذا كان خطّ الباص يتوقّف في المحطّة، خلاف ذلك، تُعيد false .

ب. معطاة مصفوفة arr من نمط BusStation ، فيها جميع محطّات الباصات في مدينة معيّنة. معلوم أنّ عددًا من خطوط الباصات تتوقّف في كلّ واحدة من المحطّات في المدينة، وبقيّة الخطوط تتوقّف فقط في جزء من المحطّات.
اكتبوا عمليّة خارجيّة باسم allStations بلغة Java أو AllStations بلغة #C ، تتلقّى المصفوفة arr .
تُعيد العمليّة سلسلة حلقات من نمط صحيح، يظهر في كلّ حلقة أحد أرقام الخطوط التي تتوقّف في **جميع** المحطّات في المدينة (كلّ خطّ كهذا يظهر مرّة واحدة فقط في السلسلة).
ملاحظة: لا أهمّيّة لترتيب الخطوط في السلسلة.

/ يتبع في صفحة 12 /

الفصل الثالث

في هذا الفصل أسئلة في ثلاثة مسارات:
ألغوريثمات، في الصفحتين 12-14.
موديلات حسابيّة، في الصفحتين 15-16.
برمجة موجّهة كائنات بلغة Java ، في الصفحات 18-21؛ برمجة موجّهة كائنات بلغة C#، في الصفحات 22-25.
اختاروا أسئلة من مسار واحد فقط.

ألغوريثمات

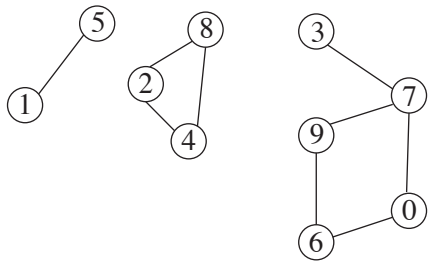
8. معطى رسم بيانيّ ليس قابلًا للرَّبْط وليس موجَّهًا $G(V, E)$ ، فيه n رؤوس، من v_0 حتّى v_{n-1} .

أ. اكتبوا ألغوريثمًا يجد ويُعيد جميع الرؤوس التي يوجد مسار بينها وبين رأس في الرسم البيانيّ – v_j .

ملاحظة: يجب كتابة ألغوريثم ناجع لا يمرّ على جميع المسارات الممكنة.

مثال: بالنسبة للرسم البيانيّ الذي أمامكم، والرأس 3 ، يُعيد الألغوريثم الرؤوس 0 ، 6 ، 7 ، 9 .

G(V, E)



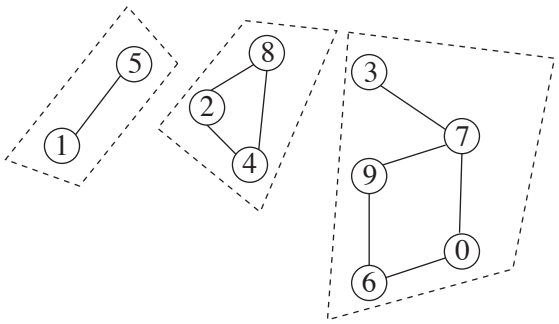
الشرح: يوجد مسار بين الرأس 3 وبين الرؤوس 0 ، 6 ، 7 ، 9 .

(انتبهوا: تكملة السؤال في الصّفحة التّالية.)

ب. "مركِّب رَبْط" في الرسم البيانيّ غير الموجَّه G(V, E) هو مجموعة رؤوس يوجد مسار بين كلّ رأسين فيها، ولا يوجد أيّ قوس يخرج من رأس في المجموعة إلى رأس ليس في المجموعة. الرأس الذي لا يوجد أيّ قوس بينه وبين أيّ رأس آخر يكون في مجموعة خاصّة به.

مثال: بالنسبة للرسم البيانيّ الذي في المثال الذي في صفحة 12، ثلاثة مركِّبات الرَّبْط مؤشَّرة بخطّ متقطِّع.

G(V, E)



اكتبوا ألغوريثمًا يجد ويُعيد مركِّب الرَّبْط الأصغر (أي المجموعة التي فيها أصغر عدد ممكن من الرؤوس) في الرسم البيانيّ G(V, E).

مثلًا: بالنسبة للمثال أعلاه، يُعيد الألغوريثم الرأسَيْن 1 ، 5 .

افترِضوا أنّه يوجد فقط مركِّب رَبْط واحد هو الأصغر.

ملاحظة: يجب كتابة ألغوريثم ناجع لا يمرّ على جميع المسارات الممكنة.

/ يتبع في صفحة 14 /

9. أمامكم ستّة ادّعاءات "أ – و". اختاروا خمسة منها، واذكروا بالنسبة لكلّ ادّعاء اخترتموه إذا كان صحيحًا أم غير صحيح.
إذا كان الادّعاء صحيحًا – علِّلوا لماذا، وإذا كان الادّعاء غير صحيح – أعطوا مثالًا يناقضه.

أ. كلّ شجرة تَنتُج من تشغيل DFS على رسم بيانيّ G ليس موجَّهًا، يمكن أن تَنتُج أيضًا من تشغيل BFS على نفس الرسم البيانيّ.

ب. كلّ شجرة تَنتُج من تشغيل DFS على رسم بيانيّ G ليس موجَّهًا كامل هي شجرة فيها لكلّ عقدة ابن واحد فقط.

جـ. معطى رسم بيانيّ موجَّه G ورأس v. إذا كان من الممكن الوصول من الرأس v إلى كلّ واحد من الرؤوس في الرسم البيانيّ، ومن الممكن أيضًا الوصول من كلّ واحد من الرؤوس إلى الرأس v، فإنّ الرسم البيانيّ G هو بالضرورة رسم بيانيّ قابل للرَّبْط جيّدًا (قويّ).

د. في الرسم البيانيّ G غير الموجَّه، إذا كان المسار الأقصر من العقدة s_j إلى العقدة s_n هو $S[s_j \ldots \mathbf{s_m} \ldots \mathbf{s_k} \ldots s_n]$، فإنّ المسار الفرعيّ من s_m وحتّى s_k هو بالضرورة المسار الأقصر من العقدة s_m إلى العقدة s_k.

هـ. الرسم البيانيّ الذي فيه دائرة لا يمكن أن يكون رسمًا بيانيًّا ذا اتّجاهَيْن.

و. لكلّ رسم بيانيّ G موزون، ليس موجَّهًا، توجد شجرة امتداديّة صغرى وحيدة.

/ يتبع في صفحة 15 /

موديلات حسابيّة

10. في هذا السؤال بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. أمامكم ادّعاءان (1)–(2) بالنسبة للُّغتَيْن L_1 ، L_2 اللّتَيْن فوق الأبجديّة $\{a, b\}$.
بالنسبة لكلّ ادّعاء، اذكروا إذا كان صحيحًا أم غير صحيح. إذا كان الادّعاء صحيحًا – علِّلوا لماذا،
وإذا كان غير صحيح – أعطوا مثالًا يُناقضه. لا علاقة بين الادّعاءَيْن.

(1) إذا كانت L_1 ، L_2 هما لغتان ليستا نظاميّتَيْن، فإنّ $L_1 \cap L_2$ هي بالضرورة لغة ليست نظاميّة.

(2) $L_1^n \cdot L_2^n$ دائمًا مساوية لِـ $(L_1 \cdot L_2)^n$.

ب. معطاة اللُّغة L فوق الأبجديّة $\{a, b, c\}$:

$L = \{w \mid w$ في لا يظهران ab ، bc التسلسلان ، $\#_a(w) + \#_c(w) =$ عدد زوجيّ $\}$

$\#_a(w)$ يُشير إلى عدد مرّات ظهور a في الكلمة w .

$\#_c(w)$ يُشير إلى عدد مرّات ظهور c في الكلمة w .

مثال لكلمة في اللُّغة L هي cba ، لأنّ مجموع مرّات ظهور الحرف a (1) والحرف c (1) هو عدد زوجيّ (2)،
والتسلسلان ab وَ bc لا يظهران في الكلمة.

(1) أمامكم خمس كلمات. بالنسبة لكلّ واحدة منها، اذكروا إذا كانت الكلمة تتبع للُّغة L أم لا.
علِّلوا إجابتكم.

aaccc , aab , ε , baa , cbbba .

(2) ابنوا أوتوماتًا نهائيًّا محدودًا ليس كاملًا يتلقّى اللُّغة L .

/ يتبع في صفحة 16 /

11. في هذا السؤال بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. أمامكم ستّ لغات فوق الأبجديّة $\{0,1\}$:

Σ^* – يُشير إلى لغة جميع الكلمات فوق الأبجديّة $\{0,1\}$.

$L_1 = \emptyset$ $\quad$ $L_4 = \{0110\}$

$L_2 = \Sigma^*$ $\quad$ $L_5 = \{\varepsilon, 110, 00, 001\}$

$L_3 = \{\varepsilon\}$ $\quad$ $L_6 = \{1, 0110, 110, 01\}$

اكتبوا اللُّغة التي تَنتُج من كلّ واحدة من العمليّات الخمس التالية:

(1) $L_5 \cap L_6$

(2) L_2^R

(3) $L_1 \cdot L_6$

(4) $L_3 \cdot L_4$

(5) $L_4 \cdot L_5$

ب. معطاة اللُّغة L فوق الأبجديّة $\{a, b, c\}$:

$$L = \left\{(ab)^k\, c^m\, b^{m+3k} \mid k, m >= 0\right\}$$

ابنوا أوتومات راصّة محدودًا يتلقّى اللُّغة L .

/ يتبع في صفحة 17 /

انتبهوا: تكملة الامتحان في الصفحة التالية.

/ يتبع في صفحة 18 /

برمجة موجّهة كائنات بلغة Java

12. في شركة لتأجير المركبات "سافروا بالسلامة"، طُوِّرَت منظومة مُحَوْسَبَة فيها الفئات التالية:
Contract – عَقْد، **Vehicle** – مركبة، **Car** – سيّارة، **Truck** – شاحنة، **Motorcycle** ـ درّاجة ناريّة.
فيما يلي تفصيل صفات الفئات:

- للفئة عَقْد (Contract) ثلاث صفات: name – اسم الزبون (نصّ)، days – عدد أيّام التأجير (من نمط صحيح)، kilo – عدد الكيلومترات التي سافرها الزبون (من نمط صحيح).
- للفئة مركبة (Vehicle) صفتان: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract).
- للفئة سيّارة (Car) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، عدد مقاعد الجلوس – seats (من نمط صحيح).
- للفئة شاحنة (Truck) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، الوزن الأقصى للحمولة – max (من نمط صحيح).
- للفئة درّاجة ناريّة (Motorcycle) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، درّاجة ناريّة للطرق الوعرة – offRoad (بولياني). إذا كانت الدرّاجة الناريّة هي للطرق الوعرة، فإنّ الصفة هي صدق، وإذا لم تكن، فإنّ الصفة هي كذب).

أ. **(1)** ارسموا مخطَّطًا هرميًّا يصف العلاقة بين فئات المنظومة المُحَوْسَبَة.
يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.

(2) اكتبوا عناوين الفئات وصفاتها. افترِضوا أنّ العمليّتين get وَ set موجودتان في جميع صفات الفئة، ولا حاجة لتطبيقهما.

ب. معطاة العمليّة البنائيّة للفئة Contract:

```
public Contract (String name, int days, int kilo)
```

لا حاجة لتطبيق العمليّة.

(1) أمامكم عنوان العمليّة البنائيّة للفئة Vehicle. تتلقّى العمليّة اسم الزبون، وعدد أيّام التأجير، وعدد الكيلومترات التي سافرها الزبون، ومميِّز المركبة.

```
public Vehicle (String name, int days, int kilo, String id)
```

طبِّقوا العمليّة البنائيّة.

(2) أمامكم عنوان العمليّة البنائيّة للفئة Car. تتلقّى العمليّة اسم الزبون، وعدد أيّام التأجير، وعدد الكيلومترات التي سافرها الزبون، ومُميِّز المركبة، وعدد مقاعد الجلوس.

```
public Car (String name, int days, int kilo, String id, int seats)
```

طبِّقوا العمليّة البنائيّة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

التسعيرة الأساسيّة التي تجبيها الشركة من تأجير مركبة – Vehicle هي 60 شيكلًا ليوم الاستئجار وشيكلان لكلّ كيلومتر سفر.

سعر استئجار سيّارة (Car) هو حسب التسعيرة الأساسيّة.

سعر استئجار شاحنة (Truck) هو حسب التسعيرة الأساسيّة، وبالإضافة إلى ذلك مبلغ لمرّة واحدة بقيمة 500 شيكل.

سعر استئجار درّاجة ناريّة (Motorcycle) هو نصف التسعيرة الأساسيّة.

جـ. العمليّة payment تُعيد عددًا حقيقيًّا مساويًا للمبلغ الذي يُطلَب من الزبون أن يدفعه مقابل كلّ واحد من أنواع المركبات التي تُؤجِّرها الشركة (حسب الكائن الذي استدعى العمليّة).

(1) اكتبوا العمليّة payment في الفئة Vehicle (كما ذُكِر أعلاه، مبلغ الدفع هو حسب التسعيرة الأساسيّة).

(2) أضيفوا العمليّة payment في الفئة/الفئات الأخرى لتنفيذ المطلوب (فقط في الفئات التي تدعو فيها الحاجة)، حسب مبادئ البرمجة الموجّهة كائنات.

ملاحظة: **لا** تستعمِلوا العمليّة instanceOf في هذا البند وعمليّات الفئة Object، ولا تُغيِّروا صفات الفئات.
الحلّ الذي يتضمّن استعمالًا لهذه العمليّات، لن يحصل على درجات.

/ يتبع في صفحة 20 /

13. أمامكم عنوانا الفئتين AA ، BB وصفاتهما.
العمليّات البنائيّة للفئتين مُشار إليها بِـ*** (يمكن أن تكون هناك أكثر من عمليّة بنائيّة واحدة للفئة) .

```
public class AA
{
    private int x;
    ***
}
public class BB extends AA
{
    private AA f;
    ***
}
```

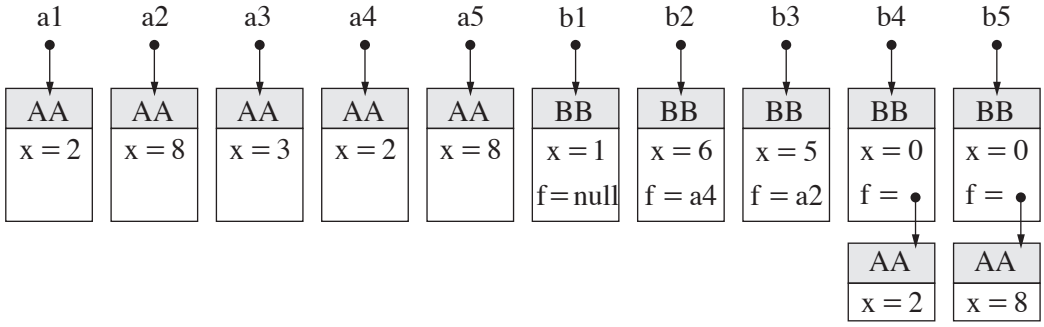
ملاحظة: انتبهوا – لكلا الفئتين لا توجد عمليّتا get وَ set .
أمامكم الفئة Test ، التي تحوي عمليّة رئيسيّة:

```
public class Test{
    public static void main(String[] args) {
        AA a1 = new AA ();
        AA a2  = new AA (8);
        AA a3 = new AA (3);
        AA a4 = new AA (a1);
        AA a5 = new AA (a2);
        BB b1 = new BB ();
        BB b2 = new BB (6, a4);
        BB b3 = new BB (5, a2);
        BB b4 = new BB (a4);
        BB b5 = new BB (a2);
    }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 21 /

أمامكم مخطَّط للكائنات التي تكوَّنت في أعقاب تشغيل قطعة كود البرمجة.
اكتبوا في الفئتين AA وَ BB العمليّات البنائيّة المطلوبة كي تَنْتُج الكائنات التي في المخطَّط.
اذكروا بالنسبة لكلّ واحد من الكائنات التي تكوَّنت، العمليّة البنائيّة الملائمة له.
ملاحظة: **لا** تُضيفوا عمليّات ليست بنائيّة في الفئتين AA وَ BB. الحلّ الذي يُضيف عمليّات ليست بنائيّة، لن يحصل على درجات.



/ يتبع في صفحة 22 /

برمجة موجّهة كائنات بلغة #C

14. في شركة لتأجير المركبات "سافروا بالسلامة"، طُوِّرَت منظومة مُحَوْسَبَة فيها الفئات التالية:
Contract – عَقْد، **Vehicle** – مركبة، **Car** – سيّارة، **Truck** – شاحنة، **Motorcycle** ـ درّاجة ناريّة.
فيما يلي تفصيل صفات الفئات:

- للفئة عَقْد (Contract) ثلاث صفات: name – اسم الزبون (نصّ)، days – عدد أيّام التأجير (من نمط صحيح)، kilo – عدد الكيلومترات التي سافرها الزبون (من نمط صحيح).
- للفئة مركبة (Vehicle) صفتان: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract).
- للفئة سيّارة (Car) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، عدد مقاعد الجلوس – seats (من نمط صحيح).
- للفئة شاحنة (Truck) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، الوزن الأقصى للحمولة – max (من نمط صحيح).
- للفئة درّاجة ناريّة (Motorcycle) ثلاث صفات: مُميِّز المركبة – id (نصّ)، عَقْد – contract (Contract)، درّاجة ناريّة للطرق الوعرة – offRoad (بولينيّ. إذا كانت الدرّاجة الناريّة هي للطرق الوعرة، فإنّ الصفة هي صدق، وإذا لم تكن، فإنّ الصفة هي كذب).

أ. **(1)** ارسموا مخطَّطًا هرميًّا يصف العلاقة بين فئات المنظومة المُحَوْسَبَة.
يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.

(2) اكتبوا عناوين الفئات وصفاتها. افترِضوا أنّ العمليّتين Get وَ Set موجودتان في جميع صفات الفئة، ولا حاجة لتطبيقهما.

ب. معطاة العمليّة البنائيّة للفئة Contract:

```
public Contract (string name, int days, int kilo)
```

لا حاجة لتطبيق العمليّة.

(1) أمامكم عنوان العمليّة البنائيّة للفئة Vehicle. تتلقّى العمليّة اسم الزبون، وعدد أيّام التأجير، وعدد الكيلومترات التي سافرها الزبون، ومميِّز المركبة.

```
public Vehicle (string name, int days, int kilo, string id)
```

طبِّقوا العمليّة البنائيّة.

(2) أمامكم عنوان العمليّة البنائيّة للفئة Car. تتلقّى العمليّة اسم الزبون، وعدد أيّام التأجير، وعدد الكيلومترات التي سافرها الزبون، ومُميِّز المركبة، وعدد مقاعد الجلوس.

```
public Car (string name, int days, int kilo, string id, int seats)
```

طبِّقوا العمليّة البنائيّة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

التسعيرة الأساسيّة التي تجبيها الشركة من تأجير مركبة – Vehicle هي 60 شيكلًا ليوم الاستئجار وشيكلان لكلّ كيلومتر سفر.

سعر استئجار سيّارة (Car) هو حسب التسعيرة الأساسيّة.

سعر استئجار شاحنة (Truck) هو حسب التسعيرة الأساسيّة، وبالإضافة إلى ذلك مبلغ لمرّة واحدة بقيمة 500 شيكل.

سعر استئجار درّاجة ناريّة (Motorcycle) هو نصف التسعيرة الأساسيّة.

جـ. العمليّة Payment تُعيد عددًا حقيقيًّا مساويًا للمبلغ الذي يُطلَب من الزبون أن يدفعه مقابل كلّ واحد من أنواع المركبات التي تُؤجِّرها الشركة (حسب الكائن الذي استدعى العمليّة).

(1) اكتبوا العمليّة Payment في الفئة Vehicle (كما ذُكِر أعلاه، مبلغ الدفع هو حسب التسعيرة الأساسيّة).

(2) أضيفوا العمليّة Payment في الفئة/الفئات الأخرى لتنفيذ المطلوب (فقط في الفئات التي تدعو فيها الحاجة)، حسب مبادئ البرمجة الموجّهة كائنات.

ملاحظة: **لا** تستعملوا العمليّتين is وَ as في هذا البند وعمليّات الفئة Object، ولا تُغيِّروا صفات الفئات.
الحلّ الذي يتضمّن استعمالًا لهذه العمليّات، لن يحصل على درجات.

/ يتبع في صفحة 24 /

15. أمامكم عنوانا الفئتين AA , BB وصفاتهما.
العمليّات البنائيّة للفئتين مُشار إليها بِـ *** (يمكن أن تكون هناك أكثر من عمليّة بنائيّة واحدة للفئة).

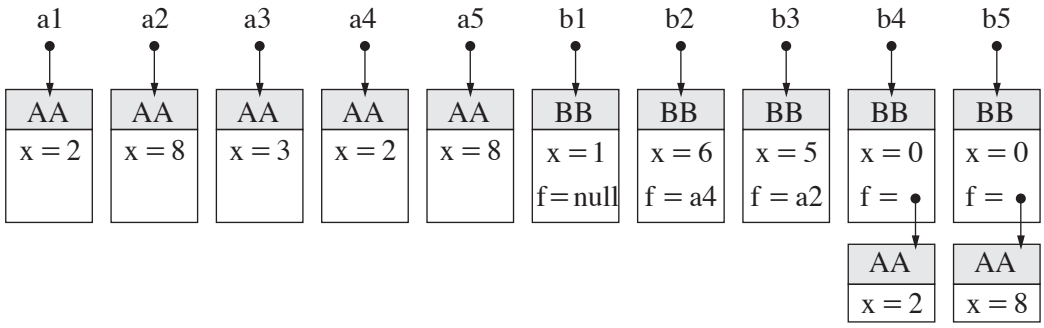
```
public class AA
{
    private int x;
    ***
}
public class BB : AA
{
    private AA f;
    ***
}
```

ملاحظة: انتبهوا – لكلا الفئتين لا توجد عمليّتا Get وَ Set .
أمامكم الفئة Test ، التي تحوي عمليّة رئيسيّة:

```
public class Test{
    public static void Main(string[] args) {
        AA a1 = new AA ();
        AA a2  = new AA (8);
        AA a3 = new AA (3);
        AA a4 = new AA (a1);
        AA a5 = new AA (a2);
        BB b1 = new BB ();
        BB b2 = new BB (6, a4);
        BB b3 = new BB (5, a2);
        BB b4 = new BB (a4);
        BB b5 = new BB (a2);
    }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

أمامكم مخطَّط للكائنات التي تكوَّنت في أعقاب تشغيل قطعة كود البرمجة.
اكتبوا في الفئتين AA وَ BB العمليّات البنائيّة المطلوبة كي تَنْتُج الكائنات التي في المخطَّط.
اذكروا بالنسبة لكلّ واحد من الكائنات التي تكوَّنت، العمليّة البنائيّة الملائمة له.
ملاحظة: **لا** تُضيفوا عمليّات ليست بنائيّة في الفئتين AA وَ BB. الحلّ الذي يُضيف عمليّات ليست بنائيّة، لن يحصل على درجات.



בהצלחה!
نتمنّى لكم النّجاح!