מדינת ישראל
משרד החינוך

دولة إسرائيل
وزارة التّربية والتّعليم

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ב, 2022
מספר השאלון: **899381**
תרגום לערבית (2)

نوع الامتحان: بجروت
موعد الامتحان: صيف 2022
رقم النّموذج: **899381**
ترجمة إلى العربيّة (2)

انتبهوا: في هذا الامتحان توجد تعليمات خاصّة.
يجب الإجابة عن الأسئلة حسب التّعليمات.

מדעי המחשב

הוראות

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שלושה פרקים.
פרק ראשון — (10x1) + (15x1) — 25 נק'
פרק שני — (25x2) — 50 נק'
פרק שלישי — (25x1) — 25 נק'
סך הכול — 100 נק'

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

ד. הוראות מיוחדות:

1. **רשמו על הכריכה החיצונית** של המחברת את **שם המסלול שלמדתם**.
המסלול הוא אחד מארבעת המסלולים האלה: מערכות מחשב ואסמבלי, 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 יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

علم الحاسوب

تعليمات

أ. مدّة الامتحان: ثلاث ساعات.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج ثلاثة فصول.
الفصل الأوّل – (10x1) + (15x1) – 25 درجة
الفصل الثّاني – (25x2) – 50 درجة
الفصل الثّالث – (25x1) – 25 درجة
المجموع – 100 درجة

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

د. تعليمات خاصّة:

1. **سجّلوا على الغلاف الخارجيّ** للدّفتر اسم **المسار الذي تعلّمتموه**.
المسار هو أحد المسارات الأربعة التالية: أنظمة حاسوب وأسمبلي، ألغوريثمات، موديلات حسابيّة، برمجة موجّهة كائنات.

2. اكتبوا جميع البرامج التي يجب كتابتها بلغة حاسوب في الفصلين الأوّل والثاني، 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

في هذا النموذج ثلاثة فصول.

يجب الإجابة عن أسئلة من ثلاثة الفصول، حسب التعليمات في كلّ فصل.

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 multiply بلغة Java أو Multiply بلغة C# تتلقّى مصفوفتين من نمط صحيح، arr1 وَ arr2، مختلفتين في كِبَرهما. نُسمّي كِبَر المصفوفة الأصغر k، ونُسمّي كِبَر المصفوفة الأكبر m.
يجب على العمليّة أن تُعيد مصفوفة جديدة من نمط صحيح بكِبَر m (كِبَر المصفوفة الكبرى من المصفوفتين arr1 وَ arr2).
يجب على المصفوفة المُعادة أن تكون حسب التفصيل التالي:
قيمة كلّ واحدة من k الخلايا الأولى، تكون مساوية لحاصل ضرب القيمتين في الخليّتين المقابلتين لهذه الخليّة في المصفوفتين arr1 وَ arr2.
بقيّة قيَم الخلايا في المصفوفة المُعادة تكون مطابقة لقيَم الخلايا المقابلة لها في المصفوفة الكبرى.

مثال: بالنسبة للمصفوفتين arr1 وَ arr2 التاليتين:

	0	1	2	3	4
arr1	1	−4	4	9	2

	0	1	2	3	4	5	6
arr2	9	2	0	−1	3	11	23

تُعيد العمليّة هذه المصفوفة:

0	1	2	3	4	5	6
9	−8	0	−9	6	11	23

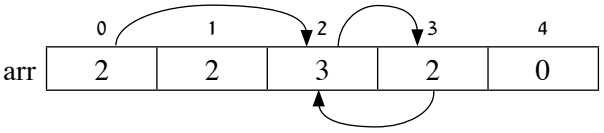
ملاحظة: افترِضوا أنّه لا توجد خلايا فارغة في المصفوفتين، وأنّ قيَم المصفوفتين سليمة.

/ يتبع في صفحة 3 /

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

2. "مسح حسب قيَم الخلايا" هو مسح مصفوفة يبدأ بالمؤشّر 0 ويمرّ إلى المؤشّر الذي رقمه مطابق لقيمة الخليّة الحاليّة التي تمّ مسحها (الخليّة في المؤشّر 0)، ومن هناك يواصل إلى الخليّة التالية بنفس الطريقة، وهكذا. إذا كانت إحدى قيَم الخلايا 0 ووصل المسح إليها، فإنّ المسح ينتهي، خلاف ذلك المسح لا ينتهي.

مثال:



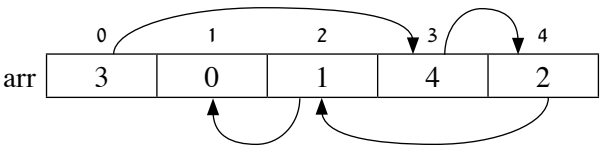
ترتيب المؤشّرات في هذا المسح هو: ... ,3 ,2 ,3 ,2 ,3 ,0 (يجب قراءة الأرقام من اليسار إلى اليمين) .

افترِضوا أنّ قيَم الخلايا لا تتعدّى حدود المصفوفة.

مصفوفة متكاملة هي مصفوفة يتحقّق فيها في "مسح حسب قيَم الخلايا" الشرط الأوّل وكذلك الشرط الثاني التاليان:

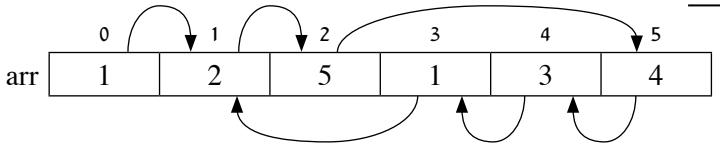
1. **جميع** خلايا المصفوفة يتمّ مسحها.
2. المسح ينتهي (لأنّه توجد في المصفوفة خليّة قيمتها 0 والمسح يصل إليها) .

مثال لمصفوفة متكاملة:



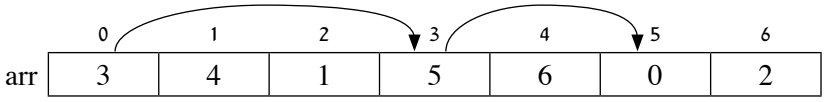
المصفوفة في هذا المثال هي مصفوفة متكاملة، لأنّ المسح يمرّ على جميع الخلايا، وينتهي.

مثال لمصفوفة **ليست** متكاملة:



هذه المصفوفة ليست متكاملة، لأنّ المسح لا ينتهي.

مثال آخر لمصفوفة **ليست** متكاملة:



هذه المصفوفة ليست متكاملة، لأنّ المسح ينتهي بدون أن تُمسَح الخلايا في المؤشّرات 1 ,2 ,4 ,6 .

طبِّقوا العمليّة المعطاة:

Java – public static boolean isPerfect (int[] arr)

C# – public static bool IsPerfect (int[] arr)

تُعيد العمليّة true إذا كانت المصفوفة arr التي نتجت هي مصفوفة متكاملة، خلاف ذلك تُعيد العمليّة false .

ملاحظة: لا يجب الحفاظ على قيَم المصفوفة.

افترِضوا أنّ قيَم الخلايا لا تتعدّى حدود المصفوفة.

/ يتبع في صفحة 4 /

3. بيكسل (Pixel) هو وحدة معلومات تصف نقطة في صورة رقميّة.
كلّ بيكسل مركَّب من دمج ثلاثة ألوان: أحمر وأخضر وأزرق.
كلّ واحد من هذه الألوان الثلاثة (أحمر وأخضر وأزرق) يتلقّى قيمة عدديّة من 0 حتّى 255 (بما في ذلك 0 وَ 255).
هذا العدد يمثّل شدّة اللون في النقطة. بواسطة دمج الألوان الثلاثة بقيَم مختلفة من 0 حتّى 255 يمكن الحصول على كلّ لون في قوس الألوان.

من أجل الحصول على نقطة لونها أحمر (بين أحمر فاتح وأحمر غامق)، يجب أن تكون قيَم الألوان:
أحمر – من **1** حتّى 255، أخضر – 0، أزرق – 0.
من أجل الحصول على نقطة لونها أخضر (بين أخضر فاتح وأخضر غامق)، يجب أن تكون قيَم الألوان:
أحمر – 0، أخضر – من **1** حتّى 255، أزرق – 0.
من أجل الحصول على نقطة لونها أزرق (بين أزرق فاتح وأزرق غامق)، يجب أن تكون قيَم الألوان:
أحمر – 0، أخضر – 0، أزرق – من **1** حتّى 255.
من أجل الحصول على نقاط ببقيّة الألوان (لا يشمل الأسود والأبيض)، فإنّ قيمتَي لونين على الأقلّ من بين الألوان أحمر وأخضر وأزرق يجب أن تكونا أكبر من 0.
من أجل الحصول على نقطة لونها أسود، فإنّ قيَم الألوان أحمر وأخضر وأزرق يجب أن تكون 0.
من أجل الحصول على نقطة لونها أبيض، فإنّ قيَم الألوان أحمر وأخضر وأزرق يجب أن تكون 255.

معطاة الفئة **Pixel** – بيكسل، ولها ثلاث صفات:

- red – لون أحمر، عدد صحيح من 0 حتّى 255.
- green – لون أخضر، عدد صحيح من 0 حتّى 255.
- blue – لون أزرق، عدد صحيح من 0 حتّى 255.

افترِضوا أنّه توجد عمليّتا get/Get وَ set/Set بالنسبة لصفات الفئة.
معطاة واجهة تطبيق جزئيّة للفئة Pixel:

وصف العمليّة	عنوان العمليّة
عمليّة بنائيّة تتلقّى قيَمًا بالنسبة لصفات الفئة.	public Pixel (int red, int green, int blue)
عمليّة تُعيد true إذا كانت النقطة بلون أحمر، خلاف ذلك تُعيد false.	**Java** – public boolean isRed () **C#** – public bool IsRed ()
عمليّة تُعيد true إذا كانت النقطة بلون أخضر، خلاف ذلك تُعيد false.	**Java** – public boolean isGreen () **C#** – public bool IsGreen ()
عمليّة تُعيد true إذا كانت النقطة بلون أزرق، خلاف ذلك تُعيد false.	**Java** – public boolean isBlue () **C#** – public bool IsBlue ()

(انتبهوا: تكملة السؤال في الصفحة التالية.) / يتبع في صفحة 5 /

أ. طبِّقوا العمليّة isRed/IsRed .
افترِضوا أنّ قيَم الصفات سليمة (من 0 حتّى 255) .

ب. في هذا البند يمكن استعمال العمليّتين isGreen / IsGreen وَ isBlue / IsBlue بدون تطبيقهما. يمكن إضافة عمليّات أخرى إلى الفئة Pixel ، لكن يجب تطبيقها.

معطاة الفئة **Structure** – ولها صفة واحدة فقط:
arr – مصفوفة أحاديّة الأبعاد من نمط **Pixel** بِكبَر N .

(1) اكتبوا عمليّة داخليّة في الفئة Structure باسم isBalanced بلغة Java أو IsBalanced بلغة C# .
تُعيد العمليّة true إذا كان عدد النقاط التي لونها أحمر في المصفوفة arr مطابقًا لعدد النقاط التي لونها أخضر ومطابقًا لعدد النقاط التي لونها أزرق. خلاف ذلك تُعيد العمليّة false .
انتبهوا: تُعيد العمليّة true أيضًا إذا كانت هناك نقاط ألوانها ليست أحمر وأخضر وأزرق، طالما أنّ أعداد النقاط الحمراء والخضراء والزرقاء متطابقة.

(2) مصفوفة "سوداء بيضاء" هي مصفوفة فيها جميع النقاط هي فقط باللونين الأسود والأبيض، وتوجد فيها على الأقلّ نقطة واحدة لونها أبيض وعلى الأقلّ نقطة واحدة لونها أسود.
اكتبوا عمليّة داخليّة في الفئة Structure باسم isBlackWhite بلغة Java أو IsBlackWhite بلغة C# . تُعيد العمليّة true إذا كانت المصفوفة arr هي مصفوفة "سوداء بيضاء"، خلاف ذلك تُعيد العمليّة false .

/ يتبع في صفحة 6 /

الفصل الثاني (50 درجة)

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

أجيبوا عن اثنين من الأسئلة 4-7 (لكلّ سؤال – 25 درجة).

4. معطاة الفئة **Range** – مجال، ولها صفتان:

- low – عدد من نمط صحيح.
- high – عدد من نمط صحيح.

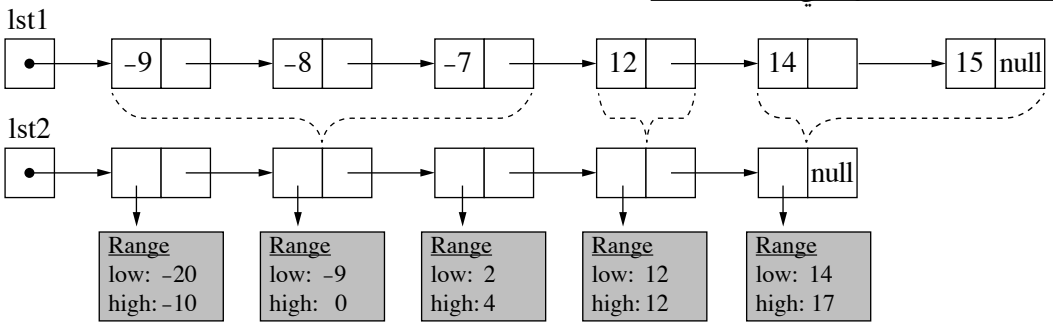
العدد high أكبر أو يساوي low (high ≥ low).

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

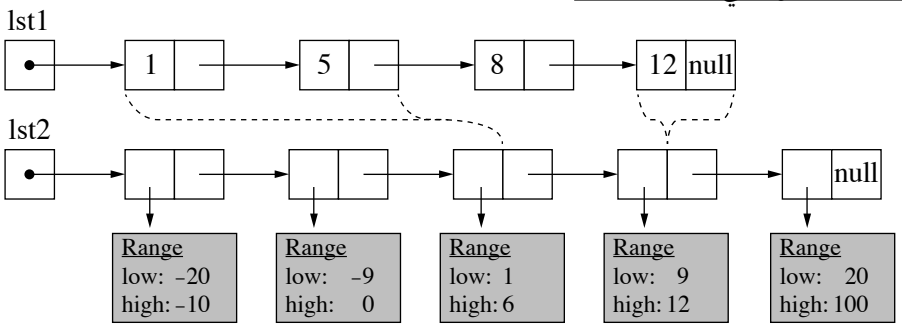
عدد ما، x ، "محويّ" في كائن من نمط Range إذا كان موجودًا في مجال الأعداد بين low وَ high (high ≥ x ≥ low).

سلسلة حلقات – lst1 من نمط صحيح "محويّة" في سلسلة حلقات – lst2 من نمط Range إذا وُجدت بالنسبة لكلّ عدد في السلسلة lst1 ، حلقة في السلسلة lst2 تحويه.

مثال لسلسلة lst1 محويّة في السلسلة lst2 :



مثال لسلسلة lst1 **ليست** محويّة في السلسلة lst2 :



شرح: العدد 8 في السلسلة lst1 ليس "محويًّا" في أيّة حلقة في السلسلة lst2 .

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

طبِّقوا العمليّة الخارجيّة التالية:

Java – public static boolean isIncluded (Node<Integer> lst1, Node<Range> lst2)

C# – public static bool IsIncluded (Node<int> lst1, Node<Range> lst2)

تُعيد العمليّة true إذا كانت lst1 "محويّة" في lst2 ، خلاف ذلك تُعيد العمليّة false . يجب أن تعمل العمليّة بزمن تعقيدات O(N) .

ملاحظة: N هو طول السلسلة الأطول من بين السلسلتين.

افتراضات:

- lst1 وَ lst2 ليستا null .
- جميع الكائنات في السلسلة lst2 التي من نمط Range ليست null .
- **السلسلة lst1 مرتّبة بترتيب تصاعديّ.**
- **السلسلة lst2 مرتّبة بترتيب تصاعديّ**، أي أنّ قيمة الـ high لكلّ حلقة أصغر من قيمة الـ low للحلقة التي تليها في السلسلة (كما يظهر في الأمثلة في الصفحة السابقة) .

/ يتبع في صفحة 8 /

5. معطاة الفئة **TwoStack** ، ولها صفتان:

- numbers – راصّة من نمط صحيح.
- sums – راصّة من نمط صحيح.

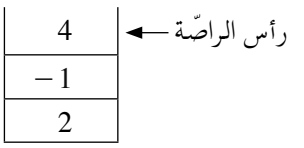
العلاقة بين الراصّة sums والراصّة numbers هي كما هو مفصّل فيما يلي:
العدد الذي في نهاية الراصّة sums يساوي العدد الذي في نهاية الراصّة numbers ،
العدد الثاني من نهاية الراصّة sums يساوي مجموع العددَيْن الأخيرَيْن في الراصّة numbers ،
العدد الثالث من نهاية الراصّة sums يساوي مجموع ثلاثة الأعداد الأخيرة في الراصّة numbers ،
وهكذا حتّى العدد الذي في رأس الراصّة sums ، الذي يساوي مجموع كلّ الأعداد في الراصّة numbers .
مثال:

أ. طبِّقوا العمليّة الداخليّة التالية:

Java – public Stack<Integer> getNums (int x)

C# – public Stack<int> GetNums (int x)

تتلقّى العمليّة عددًا x يساوي أحد الأعداد في الراصّة sums ، وتُعيد العمليّة راصّة جديدة من نمط صحيح، تظهر فيها الأعداد من الراصّة numbers التي مجموعها يساوي العدد x .
افترِضوا أنّ العدد x موجود في الراصّة sums وأنّه يظهر فيها مرّة واحدة فقط.
مثال:
بالنسبة للراصّتين اللتين في المثال أعلاه وَ x = 5 (5 هو العدد الثالث من الأسفل في الراصّة sums)، تُعيد العمليّة الراصّة التالية:



ملاحظات: – يمكن تغيير راصّتَي الفئة.
– لا توجد أهمّيّة لترتيب الأعداد في الراصّة المُعادة.
– تعريف الراصّة من نمط صحيح Stack<int>/Stack<Integer> مكافئ للتعريف StackInt .
يمكن استعمال أيّ منهما، حسب اختياركم.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 9 /

ب. طبِّقوا العمليّة الداخليّة التالية:

Java – public void eraseNum (int x)

C# – public void EraseNum (int x)

تمحو العمليّة العدد x من الراصّة numbers وتُصحِّح الراصّة sums وفقًا لذلك.
افترِضوا أنّ العدد x موجود في الراصّة numbers وأنّه يظهر فيها مرّة واحدة فقط.
ملاحظة: يجب الحفاظ على ترتيب الأعداد التي تبقّت في الراصّة numbers .

مثال:
بالنسبة للراصّتين اللتين أمامكم وَ $x = 3$:

numbers

رأس الراصّة ←	6
	−3
	3
	2

sums

رأس الراصّة ←	8
	2
	5
	2

بعد تشغيل العمليّة تبدو الراصّتان على النحو التالي:

numbers

رأس الراصّة ←	6
	−3
	2

sums

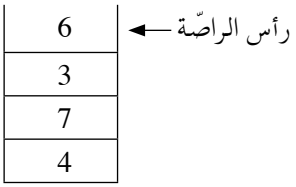
رأس الراصّة ←	5
	−1
	2

/ يتبع في صفحة 10 /

6. انتبهوا ! لهذا السؤال صيغتان : بلغة Java في الصفحتين 10-11 وبلغة #C في الصفحتين 12-13 .

للّذين يحلّون بلغة Java

أ. معطاة العمليّة stackSod1 والراصّة st :




```
public static void stackSod1(Stack<Integer> st, int element)
{
    if(st.isEmpty())
        st.push(element);
    else
    {
        int val = st.pop();
        stackSod1(st, element);
        st.push(val);
    }
}
```

ملاحظة: تعريف راصّة من نمط صحيح Stack<Integer> مكافئ للتعريف StackInt .

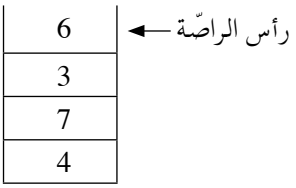
(1) ارسموا الراصّة كما ستبدو بعد استدعاء العمليّة stackSod1(st, 9) . يجب تبيين متابعة .

(2) ما هو هدف العمليّة stackSod1 ؟

(3) ما هي تعقيدات زمن تشغيل العمليّة stackSod1 ؟

(انتبهوا : تكملة السؤال في الصفحة التالية .)

ب. معطاة العمليّة stackSod2 والراصّة st :




```
public static void stackSod2 (Stack<Integer> st)
{
    if(!st.isEmpty())
    {
        int val = st.pop();
        stackSod2(st);
        stackSod1(st, val);
        st.push(val);
    }
}
```

(1) ارسموا الراصّة كما ستبدو بعد استدعاء العمليّة stackSod2(st) . يجب تبيين متابعة .

ملاحظة: في هذا البند لا حاجة لتنفيذ متابعة للعمليّة stackSod1 .

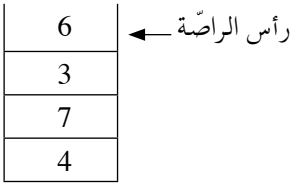
(2) ما هو هدف العمليّة stackSod2 ؟

(3) ما هي تعقيدات زمن تشغيل العمليّة stackSod2 ؟

/ يتبع في صفحة 12 /

للّذين يحلّون بلغة C#

أ. معطاة العمليّة StackSod1 والراصّة st :




```
public static void StackSod1(Stack<int> st, int element)
{
    if(st.IsEmpty())
        st.Push(element);
    else
    {
        int val = st.Pop();
        StackSod1(st, element);
        st.Push(val);
    }
}
```

ملاحظة: تعريف راصّة من نمط صحيح Stack<int> مكافئ للتعريف StackInt .

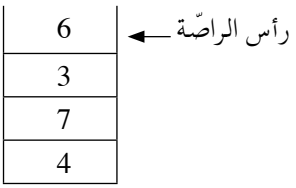
(1) ارسموا الراصّة كما ستبدو بعد استدعاء العمليّة StackSod1(st, 9) . يجب تبيين متابعة.

(2) ما هو هدف العمليّة StackSod1 ؟

(3) ما هي تعقيدات زمن تشغيل العمليّة StackSod1 ؟

(انتبهوا: تكملة السؤال في الصفحة التالية.)

ب. معطاة العمليّة StackSod2 والراصّة st :




```
public static void StackSod2 (Stack<int> st)
{
    if(!st.IsEmpty())
    {
        int val = st.Pop();
        StackSod2(st);
        StackSod1(st, val);
        st.Push(val);
    }
}
```

(1) ارسموا الراصّة كما ستبدو بعد استدعاء العمليّة StackSod2(st) . يجب تبيين متابعة.

ملاحظة: في هذا البند لا حاجة لتنفيذ متابعة للعمليّة StackSod1 .

(2) ما هو هدف العمليّة StackSod2 ؟

(3) ما هي تعقيدات زمن تشغيل العمليّة StackSod2 ؟

/ يتبع في صفحة 14 /

7. سؤال عن الشجرة البينارّية

في هذا السؤال يمكن استعمال العمليّة الخارجيّة eraseFirst / EraseFirst التالية بدون تطبيقها.

عنوان العمليّة	وصف العمليّة	أمثلة
بلغة **Java**: public static String eraseFirst (String str) بلغة **C#**: public static string EraseFirst (string str)	تُعيد العمليّة نصًّا فرعيًّا str ، بدون الرمز الأوّل. إذا كان النصّ – str فارغًا قبل استدعاء العمليّة، يكون خطأ.	– بالنسبة للنصّ "hello" = str ، تُعيد العمليّة النصّ "ello". – بالنسبة للنصّ "temp" = str ، تُعيد العمليّة النصّ "emp". – بالنسبة للنصّ "m" = str ، تُعيد العمليّة نصًّا فارغًا "". – بالنسبة للنصّ الفارغ "" = str ، يكون خطأ.

طبِّقوا العمليّة الخارجيّة التالية:

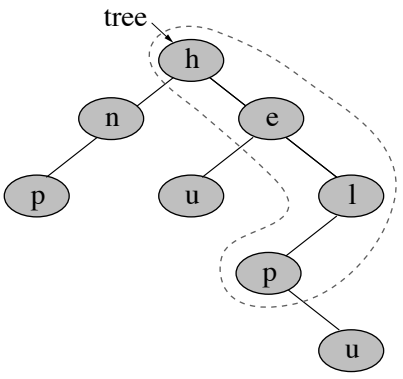
Java – public static boolean wordFromRoot (BinNode<Character> tree, String str)

C# – public static bool WordFromRoot (BinNode<char> tree, string str)

تتلقّى العمليّة نصًّا – str يحوي على الأقلّ رمزًا واحدًا، وتوجيهًا إلى شجرة رموز بينارّية – tree ليست null .
تُعيد العمليّة true إذا كان هناك مسار يبدأ بجذر الشجرة التي فيها تسلسل الرموز مطابق للنصّ – str . خلاف ذلك تُعيد العمليّة false .

ملاحظة: الحرف الصغير والحرف الكبير ليسا متطابقين.

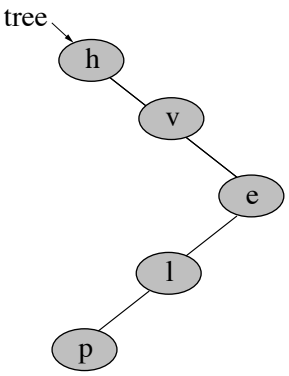
مثال:
بالنسبة للشجرة المعطاة والنصّ "help" تُعيد العمليّة true .



(انتبهوا: تكملة السؤال في الصفحة التالية.)

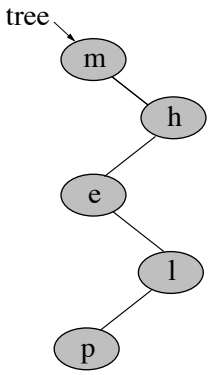
/ يتبع في صفحة 15 /

مثال:
بالنسبة للشجرة المعطاة والنصّ "help" تُعيد العمليّة false .



شرح: لا يوجد في الشجرة تسلسل رموز مطابق للنصّ "help".

مثال:
بالنسبة للشجرة المعطاة والنصّ "help" تُعيد العمليّة false .



شرح: رغم أنّه يوجد في الشجرة تسلسل رموز مطابق للنصّ "help"، تُعيد العمليّة false ، لأنّ التسلسل لا يبدأ بجذر الشجرة.

/ يتبع في صفحة 16 /

الفصل الثالث (25 درجة)

في هذا الفصل أسئلة في أربعة مسارات:
أنظمة حاسوب وأسمبلي، في الصفحات 16-18.
ألغوريثمات، في الصفحات 19-22.
موديلات حسابيّة، في الصفحة 23 .
برمجة موجّهة كائنات بلغة Java ، في الصفحات 24-27؛ برمجة موجّهة كائنات بلغة #C، في الصفحات 28-31.

أجيبوا عن سؤال واحد في المسار الذي تعلّمتموه.

أنظمة حاسوب وأسمبلي

إذا تعلّمتم هذا المسار، أجيبوا عن أحد السؤالين 8-9 (25 درجة).

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

أ. معطاة قطعة برنامج بلغة أسمبلي.

```
    MOV   SI , 30H
    MOV   CL, 4
    MOV   CH, 4
    MOV   BH, 0
A1: MOV   AX, [SI]
    MOV   BL, AL
    ROR   BL, CL
    DEC   BL
    MOV   [SI], BX
    ADD   SI, 2
    DEC   CH
    CMP   CH, 0
    JNE   A1
```

معطاة خارطة خلايا الذاكرة من 30H وحتّى 36H قبل تنفيذ قطعة البرنامج:

address	30H	32H	34H	36H
value	25H	3AH	7BH	93H

انتبهوا: خلايا الذاكرة هي بِكبَر كلمة.

(1) تتبّعوا بواسطة جدول متابعة قطعة البرنامج.
يجب أن يشمل جدول المتابعة عمودًا لكلّ واحد من الخوازن SI , CL , CH , BL , BH , AX .
بالإضافة إلى ذلك، يجب كتابة خارطة ذاكرة ملائمة كما ستبدو في نهاية قطعة البرنامج.

(2) اشرحوا ما الذي تنفّذه قطعة البرنامج.

(انتبهوا: تكملة السؤال في الصفحة التالية.) / يتبع في صفحة 17 /

ب. فيما يلي قطعة برنامج مكتوبة بلغة Java وَ C# .

```
if ( x < y + z )
{
    do {
            if ( y >= z || x < y )
                x += y;
            z --;
        }   while ( z > 0 );
}
```

المتغيّرات x وَ y وَ z هي من نمط صحيح.

اكتبوا قطعة ملائمة بلغة أسمبلي.

افترِضوا أنّ المتغيّرات x وَ y وَ z موجَّهة (signed) ومخزونة في الخوازن AX وَ BX وَ CX بالتلاؤم.

ملاحظة: لا حاجة للتطرّق إلى مسائل طفح (גלישה) ممكنة.

/ يتبع في صفحة 18 /

9. "أزواج من الجهتين" هي أزواج حدود في مصفوفة بِكبَر زوجيّ، كما هو مفصّل فيما يلي:
الزوج الأوّل مركَّب من الحدّ الأوّل ومن الحدّ الأخير في المصفوفة،
والزوج الثاني مركَّب من الحدّ الثاني ومن الحدّ قبل الأخير في المصفوفة،
والزوج الثالث مركَّب من الحدّ الثالث ومن الحدّ الذي قبل الحدّين الأخيرين في المصفوفة،
وهكذا حتّى الزوج الأخير، المركَّب من الحدّين الموجودين في منتصف المصفوفة.

"مصفوفة زوجيّة تنازليّة" هي مصفوفة بِكبَر زوجيّ تحقّق الشرط التالي:
مجموع "الأزواج من الجهتين" مرتّب بترتيب تنازليّ، أي أنّ مجموع الزوج الأوّل أكبر من مجموع الزوج الثاني، ومجموع الزوج الثاني أكبر من مجموع الزوج الثالث، وهكذا حتّى الزوج الأخير، الذي مجموعه هو الأصغر.

مثال لمصفوفة زوجيّة تنازليّة:

0	1	2	3	4	5	6	7
8	6	7	1	2	2	5	5

شرح: المصفوفة هي "زوجيّة تنازليّة"، لأنّ مجموع الزوج الأوّل (5+8) أكبر من مجموع الزوج الثاني (6+5)، الذي هو أكبر من مجموع الزوج الثالث (7+2)، الذي هو أكبر من مجموع الزوج الأخير (1+2).

مثال لمصفوفة ليست زوجيّة تنازليّة:

0	1	2	3	4	5
6	8	1	2	3	5

شرح: المصفوفة ليست "زوجيّة تنازليّة"، لأنّ مجموع الزوج الأوّل (6+5) ليس أكبر من مجموع الزوج الثاني (8+3)، وإنّما مساوٍ له.

في مقطع المعطيات عُرِّفت المعطيات التالية:

```
ARR DB 20 DUP (?)
REZ  DB ?
```

اكتبوا قطعة برنامج تفحص إذا كانت المصفوفة ARR هي مصفوفة زوجيّة تنازليّة.
إذا كانت كذلك، تُخزِّن قطعة البرنامج العدد 1 في المتغيّر REZ، خلاف ذلك تُخزِّن فيه العدد 0.
ملاحظة: لا حاجة لفحص أنّ المصفوفة بِكبَر زوجيّ.

/ يتبع في صفحة 19 /

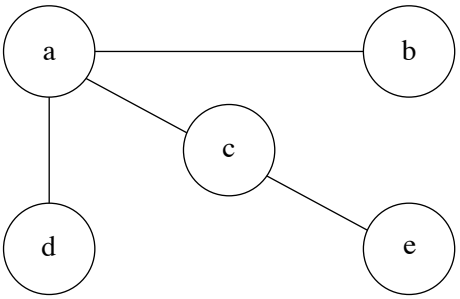
ألغوريثمات

إذا تعلّمتم هذا المسار، أجيبوا عن أحد السؤالين 10–11 (25 درجة) .

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

أ. فيما يلي رسم بيانيّ $G = (V, E)$ ليس موجَّهًا:

الرسم البيانيّ G



(1) هل الرسم البيانيّ قابل للربط؟ علِّلوا.

(2) هل الرسم البيانيّ ثنائيّ الجهات؟ علِّلوا.

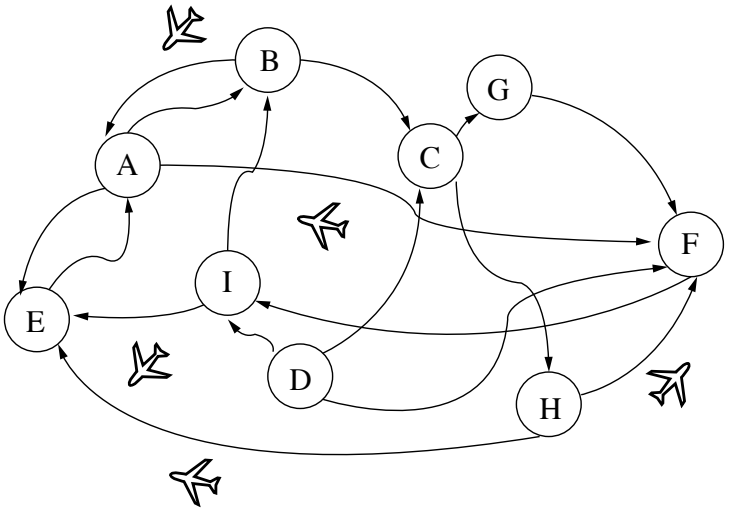
(3) هل الرسم البيانيّ كامل (تامّ)؟ علِّلوا.

(4) هل الرسم البيانيّ هو شجرة؟ علِّلوا.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 20 /

ب. تُشغِّل شركة طيران في الأيّام العاديّة خطوط طيران بين الدول A ,B ,C ,D ,E ,F ,G ,H ,I حسب الرسم التالي:



(1) ارسموا قائمة المجاوَرات التي تمثّل خطوط الطيران بين الدول.

في أعقاب تغييرات في شركة الطيران، تقرّر إلغاء عدّة خطوط طيران في الدول التالية:
الدولة G – إلغاء الرحلات القادمة والرحلات المغادرة.
الدولة H – إلغاء الرحلات المغادرة إلى الدولة F.
الدولة D – إلغاء الرحلات المغادرة إلى الدولة C.
الدولة A – إلغاء الرحلات المغادرة إلى الدولة E.

(2) ارسموا قائمة المجاوَرات التي تنتج بعد الإلغاءات المذكورة أعلاه.

(3) I. ترغب آمال في الوصول من دولة معيّنة إلى دولة أخرى.
لهذا الغرض تريد أن تفحص إذا كانت تستطيع الوصول إلى تلك الدولة، وإذا كانت تستطيع بالفعل، ترغب آمال في إيجاد مسار طيران ما يصل إلى تلك الدولة.
ما هو الألغوريثم الذي عليها استعماله لهذه الأغراض؟ اشرحوا.
ملاحظة: إذا كانت هناك عدّة ألغوريثمات ملائمة، اختاروا واحدًا منها.

II. فعِّلوا الألغوريثم الذي كتبتم عنه في البند I(3) من أجل إيجاد مسار الطيران من الدولة A إلى الدولة E . اكتبوا مسار الطيران الذي وجدتموه.
يجب تنفيذ متابعة مفصَّلة في كلّ مرحلة وفقًا للألغوريثم الذي فعّلتموه.

/ يتبع في صفحة 21 /

11. في هذا السؤال أربعة بنود، "أ–د"، لا علاقة بينها. أجيبوا عن جميع البنود.

أ. الرسالة: It's my bank account حُوِّلت إلى بِتّات، باستثناء الكلمة my .
حَوِّلوا الكلمة my إلى بِتّات.
القيمة الـAscii للحرف m هي 109 بأساس عشريّ، والقيمة الـAscii للحرف y هي 121 بأساس عشريّ.

ب. فيما يلي مجموعة بِتّات – m ، تمثّل القيمة الـ Ascii للكلمة Hey ، ومجموعة بِتّات – key ، التي هي لوحة لمرّة واحدة (OTP).

m – 010010000110010101111001

key – 110110100100010111101101

شَفِّروا الكلمة Hey بواسطة العمليّة xor مع key .

جـ. شفَّرت رنا ويزيد رسالة ما – m ، طولها n هو زوجيّ، بواسطة العمليّة xor مع مفتاح – key ، وحصلا على رسالة مشفَّرة– cipher . بغرض الحذر، نصّفت رنا ويزيد key إلى نصفين. النصف الأوّل للمفتاح، key_1 (الذي طوله n/2)، أخذَتْه رنا معها إلى مكان معيّن، والنصف الآخر – key_2 (الذي طوله n/2)، أخذه يزيد إلى مكان آخر.
بغرض فكّ تشفير الرسالة، نفّذت صديقتهما هالة الأعمال التالية حسب الترتيب:

- أخذت هالة من يزيد key_2 ، وأضافت إلى بدايته كمّيّة أصفار طولها n/2 (وبذلك أصبح طول key_2 بطول n).
- نفّذت هالة cipher xor key_2 وحصلت على مجموعة بِتّات. نُسمّي المجموعة التي حصلت عليها halfMessage .

- أخذت هالة من رنا key_1 ، وأضافت إلى آخره كمّيّة أصفار طولها n/2 (وبذلك أصبح طول key_1 بطول n).
- نفّذت هالة halfMessagae xor key_1 وحصلت على مجموعة بِتّات. نُسمّي المجموعة التي حصلت عليها fullMessage .

اكتبوا هل فكّ التشفير سليم (أي، هل fullMessage هي الرسالة m). علِّلوا.

ملاحظة: إجابة بدون تعليل لا تحصل على درجات.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 22 /

د. استعمل عمرو وزيد العمليّة CreatePrg/createPrg بهدف تكوين مفتاح PRG ، الذي شَفَّرا به رسالة بواسطة العمليّة xor . العمليّة تتلقّى نصّ بِتّات – seed ، وطول الرسالة المُراد تشفيرها – len، وتُعيد لوحة لمرّة واحدة (OTP).

العمليّة CreatePrg/createPrg مكتوبة أمامكم بلغة Java وبلغة C# .

مريم تعرف ما هو البِتّ الأخير للـseed . هل تستطيع مريم فكّ تشفير الرسالة في محاولة واحدة؟ علِّلوا.

ملاحظة: إجابة بدون تعليل لا تحصل على درجات.

بلغة **C#** :	بلغة **Java** :

public static string CreatePrg (string seed, int len) {
  string str;
  if (seed[seed.Length -1] == '1')
  str="0";
  else
  str="1";
  for (int i = 1; i < len; i ++) {
  if (str[str.Length -1] == '0')
  str += "1";
  else
  str += "0";
  }
  return str;
}
	public static String createPrg (String seed, int len) {
	  String str;
	  if (seed.charAt(seed.length() -1) == '1')
	  str="0";
	  else
	  str="1";
	  for (int i = 1; i < len; i ++) {
	  if (str.charAt(str.length() -1) == '0')
	  str += "1";
	  else
	  str += "0";
	  }
	  return str;
	}

/ يتبع في صفحة 23 /

موديلات حسابيّة

إذا تعلّمتم هذا المسار، أجيبوا عن أحد السؤالين 12–13 (25 درجة) .

12. معطاة اللغة L فوق الأبجديّة $\{a, b\}$:

$$L = \{a^n b^m \mid n > 0, m >= 0, m \% 4 = n \% 2\}$$

أ. أمامكم 10 كلمات، بالنسبة لكلّ واحدة منها، اذكروا هل الكلمة تتبع لِلُّغة L ، وعلِّلوا.

a, ab, aaabbb, aaabbbbb, aabb, aaab, aa, bbbb, aaaa, abbb

ب. ابنوا أوتوماتًا نهائيًّا كاملًا يتلقّى اللغة L .

13. معطاة اللغة L_1 فوق الأبجديّة $\{a, b, c\}$:

$$L_1 = \{c^{1+k+n} b^k a^{2n} \mid n, k \geq 1\}$$

أ. **(1)** ما هي أقصر كلمة في اللغة L_1 ؟

(2) ابنوا أوتومات راصّة يتلقّى اللغة L_1 .

ب. نُعرِّف اللغة L_2 فوق الأبجديّة $\{a, b, c, d\}$:

$$L_2 = L_1 \cdot d \cdot R(L_1)$$

بالنسبة لكلّ واحدة من الكلمات التي أمامكم، اذكروا هل الكلمة تتبع لِلُّغة L_2 ، وعلِّلوا.

cccbaaddaabccc

cccbaadcccbaa

cccbaadaaaabcccc

ملاحظة: لا حاجة لبناء أوتومات بالنسبة لِلُّغة L_2 .

/ يتبع في صفحة 24 /

برمجة موجّهة كائنات بلغة Java

إذا تعلّمتم هذا المسار وتكتبون بلغة Java، أجيبوا عن أحد السؤالين 14-15. (25 درجة)

14. يرغب صاحب دكّان كلّ شيء في بيع منتَجاته في الإنترنت (أونلاين). قرّر صاحب الدكّان في المرحلة الأولى بيع قمصان وبنطلونات فقط في الإنترنت.

لهذا الغرض بُنِيَت الفئات التالية:

– الفئة **Clothing** ، التي تمثّل لباسًا.

– الفئة **Shirt** ، التي تمثّل قميصًا.

– الفئة **Pants** ، التي تمثّل بنطلونًا.

```
public class Clothing {
    private int id;              // رقم تشخيص للمنتَج
    private String fabric;       // نوع القماش
    private String color;        // لون اللباس
    private double price;        // السعر
}
public class Shirt extends Clothing {
    private String size;         // المقاس (S, M, L, XL, XXL)
}
public class Pants extends Clothing {
    private int length;          // الطول بالإنشات
    private int width;           // العرض بالإنشات
}
```

افترِضوا أنّه توجد عمليّتا get وَ set بالنسبة للصفات في جميع الفئات.

أ. الفئة **Cart** تمثّل سلّة مشتريات. للفئة ثلاث صفات:

name – اسم المشتري.

arr – مصفوفة تحوي قمصانًا وبنطلونات أيضًا.

current – عدد المنتَجات الحاليّ في الشروة.

(1) اكتبوا عنوان الفئة Cart وصفاتها.

(2) ارسموا مخطّطًا هرميًّا يصف العلاقة بين الفئات: Cart , Pants , Shirts , Clothing.

يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 25 /

ב. في المرحلة الثانية، قرّر صاحب الدكّان بيع كتب أيضًا في الإنترنت.
أمامكم الفئة **Book** التي تمثّل كتابًا.

```
public class Book {
    private int id;                 // رقم تشخيص للمنتَج
    private String bookName;        // اسم الكتاب
    private String author;          // اسم المؤلِّف
    private double price;           // السعر
}
```

افترِضوا أنّه توجد عمليّتا get وَ set بالنسبة لصفات الفئة.
كي تستطيع المصفوفة arr في الفئة Cart أن تحوي كتبًا أيضًا، نُفِّذت الأعمال التالية:

- عُرِّفت الفئة **Product** – منتَج.
- تمّ تغيير نمط المتغيّر arr في الفئة Cart إلى Product.
- كُتبت في الفئات الملائمة (بما فيها الفئة Book) علاقات توريث جديدة بينها وبين الفئة Product.
- جزء من الصفات التي ظهرت حتّى الآن في عدد من الفئات الأخرى (بما فيها الفئة Book) نُقِلَت إلى الفئة Product، وفقًا لمبادئ البرمجة الموجّهة كائنات (لا توجد صفات إضافيّة في الفئة Product باستثناء هذه الصفات).

(1) ارسموا مخطّطًا هرميًّا معدَّلًا يصف العلاقة بين الفئات:
Cart , Product , Book , Pants , Shirt , Clothing .
يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆ .

(2) اكتبوا عنوان الفئة Product وصفاتها.

(3) اكتبوا من جديد الفئات التي طرأ فيها تغيير في أعقاب إضافة الفئة Product .
انتبهوا: يجب التطرّق إلى علاقات توريث جديدة وإلى التغيير الذي طرأ في صفات الفئة أيضًا.

ג. في المرحلة الأخيرة، قرّر صاحب الدكّان الإعلان عن حملة مؤقّتة: في الشراء في الإنترنت، تُباع جميع المنتَجات بتخفيض 10%، وبالنسبة لشراء الملابس (القمصان والبنطلونات) يكون تخفيض إضافيّ مقداره 10 شواكل (افترِضوا أنّ السعر بقي موجبًا بعد التخفيض).

معطاة العمليّة الداخليّة في الفئة Product:

```
public double getDiscountPrice()
```

استدعاء العمليّة يُعيد سعر المنتَج (قميص/بنطلون/كتاب) بعد التخفيض – وفقًا لكلّ منتَج.
طبِّقوا العمليّة في الفئة Product ، وأَضيفوا عمليّة/عمليّات في الفئة/الفئات الملائمة لتنفيذ المطلوب.
بالنسبة لكلّ عمليّة أضفتموها، اذكروا في أيّة فئة أُضيفت.
لا تستعملوا العمليّة instanceof وعمليّات الفئة Object ولا تُغيِّروا صفات الفئات.

/ يتبع في صفحة 26 /

15. معطاة الفئات الثلاث – Mammal (ثدييّ) وَ Antelope (ظبي) وَ Beaver (قندس) :

```
public class Mammal   {
      protected int weight;

      public Mammal (int w) {
            weight = w;
      }
      public int getWeight () {
            return weight;
      }
      public boolean isSame (Mammal other) {
            System.out.println ("In Mammal");
            return (this == other);
      }
}
```

.....

```
public class Antelope extends Mammal   {
      public Antelope (int w) { super (w); }
      public boolean isSame(Antelope other) {
            System.out.println ("In Antelope");
            return ((other != null)  && (this.weight == other.weight));
      }
}
```

.....

```
public class Beaver extends Mammal   {
      public Beaver (int w) { super (w); }
      public boolean isSame (Mammal  other) {
            System.out.println ("In Beaver");
            return ((other != null)  && (other instanceof Beaver)  && (this.weight == ((Beaver)other).weight));
      }
}
```

.....

(انتبهوا : تكملة السؤال في الصفحة التالية.)
/ يتبع في صفحة 27 /

معطاة الفئة Program :

```
public class Program {
    public static void main (String[] args) {
        Antelope a1 = new Antelope (10);
        Object a2 = new Antelope (10);
        Beaver b1 = new Beaver (10);
        Mammal b2 = new Beaver (10);
        *******
    }
}
```

الفئة Program موجودة في رزمة (Package) تختلف عن الفئة Mammal .

عوِّضوا (ضعوا) كلّ واحد من الأسطر 1–10 التالية في العمليّة main ، في المكان المُشار إليه أعلاه بنجمات ******* . اكتبوا في الدفتر رقم السطر، واذكروا هل الكود سليم أم غير سليم. إذا كان الكود سليمًا – اكتبوا المُخرَج، وإذا كان غير سليم – فسِّروا لماذا.

1. System.out.println (a1.weight);
2. System.out.println (((Beaver)a2).getWeight());
3. System.out.println (a1.isSame (a2));
4. System.out.println (a2.isSame (a1));
5. System.out.println (b1.isSame (b2));
6. System.out.println (b2.isSame (b1));
7. System.out.println (a1.isSame ((Beaver)b2));
8. System.out.println (a1.isSame ((Antelope)a2));
9. System.out.println (b1.isSame ((Antelope)a2));
10. System.out.println (b1.isSame ((Beaver)a2));

/ يتبع في صفحة 28 /

برمجة موجّهة كائنات بلغة #C

إذا تعلّمتم هذا المسار وتكتبون بلغة #C، أجيبوا عن أحد السؤالين 16-17 (25 درجة).

16. يرغب صاحب دكّان كلّ شيء في بيع منتَجاته في الإنترنت (أونلاين). قرّر صاحب الدكّان في المرحلة الأولى بيع قمصان وبنطلونات فقط في الإنترنت.

لهذا الغرض بُنِيَت الفئات التالية:

– الفئة **Clothing** ، التي تمثّل لباسًا.

– الفئة **Shirt** ، التي تمثّل قميصًا.

– الفئة **Pants** ، التي تمثّل بنطلونًا.

```
public class Clothing {
    private int id;              // رقم تشخيص للمنتَج
    private string fabric;       // نوع القماش
    private string color;        // لون اللباس
    private double price;        // السعر
}
public class Shirt : Clothing {
    private string size;         // المقاس (S, M, L, XL, XXL)
}
public class Pants : Clothing {
    private int length;          // الطول بالإنشات
    private int width;           // العرض بالإنشات
}
```

افترِضوا أنّه توجد عمليّتا Get وَ Set بالنسبة للصفات في جميع الفئات.

أ. الفئة **Cart** تمثّل سلّة مشتريات. للفئة ثلاث صفات:

name – اسم المشتري.

arr – مصفوفة تحوي قمصانًا وبنطلونات أيضًا.

current – عدد المنتَجات الحاليّ في الشروة.

(1) اكتبوا عنوان الفئة Cart وصفاتها.

(2) ارسموا مخطّطًا هرميًّا يصف العلاقة بين الفئات: Cart , Pants , Shirts , Clothing.

يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 29 /

ב. في المرحلة الثانية، قرّر صاحب الدكّان بيع كتب أيضًا في الإنترنت.
أمامكم الفئة **Book** التي تمثّل كتابًا.

```
public class Book {
    private int id;                 // رقم تشخيص للمنتَج
    private string bookName;        // اسم الكتاب
    private string author;          // اسم المؤلِّف
    private double price;           // السعر
}
```

افترِضوا أنّه توجد عمليّتا Get وَ Set بالنسبة لصفات الفئة.
كي تستطيع المصفوفة arr في الفئة Cart أن تحوي كتبًا أيضًا، نُفِّذت الأعمال التالية:

- عُرِّفت الفئة **Product** – منتَج.
- تمّ تغيير نمط المتغيّر arr في الفئة Cart إلى Product.
- كُتبت في الفئات الملائمة (بما فيها الفئة Book) علاقات توريث جديدة بينها وبين الفئة Product.
- جزء من الصفات التي ظهرت حتّى الآن في عدد من الفئات الأخرى (بما فيها الفئة Book) نُقِلَت إلى الفئة Product، وفقًا لمبادئ البرمجة الموجّهة كائنات (لا توجد صفات إضافيّة في الفئة Product باستثناء هذه الصفات).

(1) ارسموا مخطّطًا هرميًّا معدَّلًا يصف العلاقة بين الفئات:
Cart , Product , Book , Pants , Shirt , Clothing.
يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.

(2) اكتبوا عنوان الفئة Product وصفاتها.

(3) اكتبوا من جديد الفئات التي طرأ فيها تغيير في أعقاب إضافة الفئة Product.
انتبهوا: يجب التطرّق إلى علاقات توريث جديدة وإلى التغيير الذي طرأ في صفات الفئة أيضًا.

ג. في المرحلة الأخيرة، قرّر صاحب الدكّان الإعلان عن حملة مؤقّتة: في الشراء في الإنترنت، تُباع جميع المنتَجات بتخفيض 10%، وبالنسبة لشراء الملابس (القمصان والبنطلونات) يكون تخفيض إضافيّ مقداره 10 شواكل (افترِضوا أنّ السعر بقي موجبًا بعد التخفيض).

معطاة العمليّة الداخليّة في الفئة Product:

```
public virtual double GetDiscountPrice()
```

استدعاء العمليّة يُعيد سعر المنتَج (قميص/بنطلون/كتاب) بعد التخفيض – وفقًا لكلّ منتَج.
طبِّقوا العمليّة في الفئة Product، وأَضيفوا عمليّة/عمليّات في الفئة/الفئات الملائمة لتنفيذ المطلوب.
بالنسبة لكلّ عمليّة أضفتموها، اذكروا في أيّة فئة أُضيفت.
لا تستعملوا العمليّتين is وَ as وعمليّات الفئة Object ولا تُغيِّروا صفات الفئات.

/ يتبع في صفحة 30 /

17. معطاة الفئات الثلاث – Mammal (ثدييّ) وَ Antelope (ظبي) وَ Beaver (قندس) :

```
public class Mammal   {
      protected int weight;

      public Mammal (int w) {
            weight = w;
      }
      public int GetWeight () {
            return weight;
      }
      public virtual bool IsSame (Mammal other) {
            Console.WriteLine ("In Mammal");
            return (this == other);
      }
}
```

.....

```
public class Antelope : Mammal   {
      public Antelope (int w) : base (w) {}
      public bool IsSame(Antelope other) {
            Console.WriteLine ("In Antelope");
            return ((other != null)  && (this.weight == other.weight));
      }
}
```

.....

```
public class Beaver : Mammal   {
      public Beaver (int w) : base (w) {}
      public override bool IsSame (Mammal  other) {
            Console.WriteLine ("In Beaver");
            return ((other != null)  && (other is Beaver)  && (this.weight == ((Beaver)other).weight));
      }
}
```

.....

(انتبهوا : تكملة السؤال في الصفحة التالية .)
/ يتبع في صفحة 31 /

معطاة الفئة Program :

```
public class Program {
    public static void Main (string[] args) {
        Antelope a1 = new Antelope (10);
        Object a2 = new Antelope (10);
        Beaver b1 = new Beaver (10);
        Mammal b2 = new Beaver (10);
        *******
    }
}
```

عوِّضوا (ضعوا) كلّ واحد من الأسطر 1–10 التالية في العمليّة Main ، في المكان المُشار إليه أعلاه بنجمات *******.
اكتبوا في الدفتر رقم السطر، واذكروا هل الكود سليم أم غير سليم. إذا كان الكود سليمًا – اكتبوا المُخرَج، وإذا كان غير سليم – فسِّروا لماذا.

1. Console.WriteLine (a1.weight);
2. Console.WriteLine (((Beaver)a2).GetWeight());
3. Console.WriteLine (a1.IsSame (a2));
4. Console.WriteLine (a2.IsSame (a1));
5. Console.WriteLine (b1.IsSame (b2));
6. Console.WriteLine (b2.IsSame (b1));
7. Console.WriteLine (a1.IsSame ((Beaver)b2));
8. Console.WriteLine (a1.IsSame ((Antelope)a2));
9. Console.WriteLine (b1.IsSame ((Antelope)a2));
10. Console.WriteLine (b1.IsSame ((Beaver)a2));

בהצלחה!
نتمنّى لكم النّجاح!

זכות היוצרים שמורה למדינת ישראל.
אין להעתיק או לפרסם אלא ברשות משרד החינוך.
حقوق الطّبع محفوظة لدولة إسرائيل.
النّسخ أو النّشر ممنوعان إلّا بإذن من وزارة التّربية والتّعليم.