מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ה, 2025
מספר השאלון: 899271
תרגום לערבית (2)

מדעי המחשב

הוראות

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שני פרקים.

פרק ראשון	(25x2)	— 50	נק'
פרק שני	(25x2)	— 50	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 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 יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: صيف 2025
رقم النّموذج: 899271
ترجمة إلى العربيّة (2)

علم الحاسوب

تعليمات

أ. مدّة الامتحان: ثلاث ساعات.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج فصلان.

الفصل الأوّل	(25x2)	– 50	درجة
الفصل الثّاني	(25x2)	– 50	درجة
المجموع		– 100	درجة

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

د. تعليمات خاصّة:
1. **اكتبوا على الغلاف الخارجيّ** للدّفتر **اسم المسار الّذي تعلّمتموه.**
المسار هو أحد المسارات الثّلاثة التّالية:
ألغوريثمات، موديلات حسابيّة،
برمجة موجّهة كائنات.
2. اكتبوا جميع البرامج الّتي 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

في هذا النموذج فصلان. يجب الإجابة عن أسئلة من الفصلين، حسب التعليمات في كلّ فصل.

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

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

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

الفصل الأوّل (50 درجة)

أجيبوا عن اثنين من الأسئلة 1-3. (لكلّ سؤال – 25 درجة.)

1. معطاة الفئة **Game** – لعبة حاسوب، ولها صفتان:

- name – اسم اللعبة، من نمط نصّ.
- price – سعر اللعبة – عدد أكبر من 0، من نمط صحيح.

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

معطاة الفئة **Store** – متجر ألعاب حاسوب، ولها صفة واحدة:

- lst – توجيه إلى سلسلة حلقات ليست فارغة، من نمط **Game**. كلّ حلقة في السلسلة تحوي لعبة تُباع في المتجر.

ملاحظة: الألعاب ليست مرتّبة في السلسلة بترتيب معيّن، وكلّ لعبة تظهر مرّة واحدة فقط.

أ. طبِّقوا عمليّة واجهة تطبيق الفئة **Store** التي أمامكم:

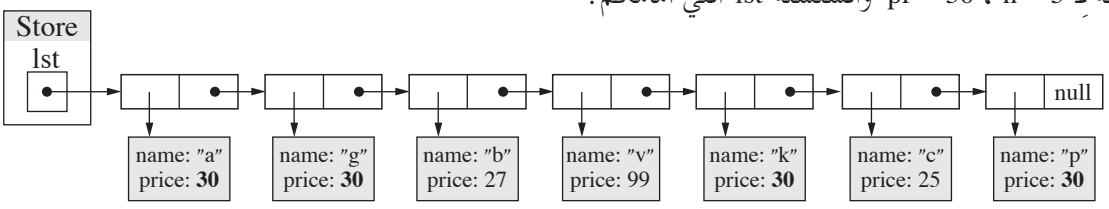
Java – public int remove (int n, int pr)

C# – public int Remove (int n, int pr)

تمحو العمليّة من السلسلة n ألعاب سعر كلّ منها pr. إذا كان هناك أكثر من n ألعاب سعر كلّ منها pr، تُمحَى فقط n الألعاب الأولى من بينها. إذا كان هناك أقلّ من n ألعاب سعر كلّ منها pr، تُمحَى هذه الألعاب فقط.
تُعيد العمليّة عدد الألعاب التي مُحِيَت (أي على الأكثر n، لكن يمكن أن يكون أقلّ).
افترِضوا أنّ n وَ pr أكبر من 0.

ملاحظة: بقيّة الألعاب في السلسلة تبقى بنفس الترتيب. إذا لم يكن في السلسلة أيّة لعبة سعرها pr، تبقى السلسلة بدون أيّ تغيير، وتُعيد العمليّة 0.

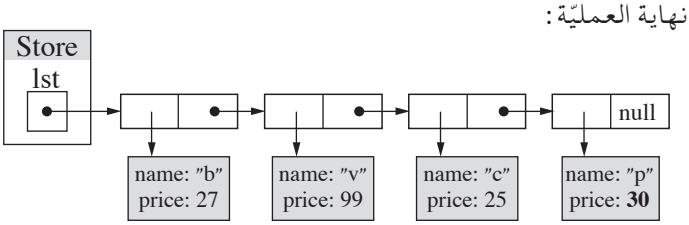
مثال: بالنسبة لِـ n = 3، pr = 30 والسلسلة lst التي أمامكم:



(انتبهوا: تكملة السؤال في الصفحة التالية.)

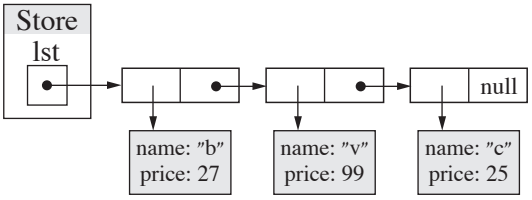
/ يتبع في صفحة 3 /

تُعيد العمليّة 3، وتبدو السلسلة على النحو التالي في نهاية العمليّة:



الشرح: يوجد في السلسلة أربع ألعاب سعر كلّ منها 30 . لأنّ $n = 3$ ، مُحِيَت الألعاب الثلاث الأولى في السلسلة التي سعر كلّ منها 30، وأعادت العمليّة 3 .

مثال إضافيّ: بالنسبة لنفس السلسلة من المثال الأصليّ أعلاه وَ $\mathbf{n = 5}$ ، $pr = 30$ ، تُعيد العمليّة 4 ، وتبدو السلسلة على النحو التالي في نهاية العمليّة:



الشرح: يوجد فقط أربع ألعاب سعر كلّ منها 30 . لذلك مُحِيَت من السلسلة الألعاب الأربع التي سعر كلّ منها 30 ، وأعادت العمليّة 4 .

ب. طبِّقوا عمليّة واجهة تطبيق الفئة **Store** التي أمامكم:

Java – public int removeCheap (int num)

C# – public int RemoveCheap (int num)

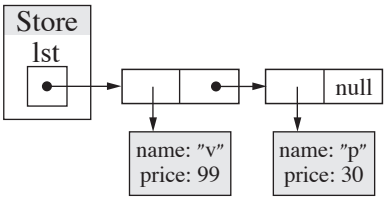
تمحو العمليّة من السلسلة الألعاب الـ num الأرخص. تُعيد العمليّة المجموع الكلّيّ لأسعار جميع الألعاب التي مُحِيَت.

افترِضوا أنّ num أكبر من 0 وأصغر من عدد الألعاب في السلسلة.

ملاحظات:

- الألعاب التي لم تُمْحَ في السلسلة، تبقى بنفس الترتيب.
- إذا كان عدد الألعاب التي سعرها متساوٍ أكبر من عدد الألعاب التي يجب أن تُمحَى، ليس هناك أهمّيّة لأيّ منها يُمحَى.
- يمكن استعمال العمليّة التي في البند "أ".

مثال: بالنسبة لنفس السلسلة من المثال الأصليّ في البند "أ" وَ $num = 5$ ، تُعيد العمليّة 142 ، وتبدو السلسلة على النحو التالي:



الشرح: الألعاب الخمس الأرخص (25+27+30+30+30) مُحِيَت من السلسلة، والمجموع الكلّيّ لأسعارها هو 142 . بقيت في السلسلة لعبة واحدة سعرها 99 ، وواحدة سعرها 30 (يمكن إبقاء لعبة مختلفة سعرها 30 في السلسلة، ليس هناك أهمّيّة لأيّ منها تبقى).

/ يتبع في صفحة 4 /

2. انتبهوا: لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 4-5 ، وبلغة **C#** في الصفحتين 6-7 .

للّذين يحلُّون بلغة Java

أ. أمامكم العمليّة mmm ، التي تتلقّى دَوْرًا – q فيه أعداد أكبر من 0 ، وعددًا صحيحًا – z أكبر من 0 .

```
public static boolean mmm (Queue<Integer> q, int z)
{
    q.insert (0);
    int num = q.head();
    int y = 0;
    while (q.head() > 0)
    {
        if (y < z)
        {
            if (q.head() == num)
            {
                y++;
            }
            else
            {
                num = q.head();
                y = 1;
            }
        }
        q.insert (q.remove());
    }
    q.remove();
    return y == z;
}
```

معطى الدَّوْر q من نمط صحيح:

رأس الدَّوْر							نهاية الدَّوْر
	1	3	1	1	1	2	

(1) تتبَّعوا، بواسطة جدول المتابعة الذي أمامكم العمليّة mmm(q , 4) ، واكتبوا ماذا تُعيد العمليّة.

الدَّوْر q	num	y	y < z	q.head() == num

(2) اشرحوا ماذا تنفّذ العمليّة mmm .

(3) ما هي تعقيدات زمن تشغيل العمليّة mmm ؟ علِّلوا إجابتكم.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 5 /

ب. أمامكم العمليّة what ، التي تتلقّى دَوْرًا – q فيه أعداد أكبر من 0 ، وكِبَر الدَّوْر – n .

```
public static int what (Queue<Integer> q, int n)
{
    if (mmm (q, n))
        return n;
    return what (q, n – 1);
}
```

معطى الدَّوْر – q من نمط صحيح:

رأس الدَّوْر							نهاية الدَّوْر
	1	3	1	1	1	2	

(1) تتبَّعوا العمليّة what (q, 6) ، واكتبوا ماذا تُعيد العمليّة (لا حاجة لمتابعة العمليّة mmm) .
تحوي المتابعة، في كلّ استدعاء، قيمتَيْ q ، n والقيمة المُعادة.
أمامكم اقتراح لجدول متابعة (استعمال هذا الجدول ليس إلزاميًّا) .

الدَّوْر q المتلقَّى في العمليّة	قيمة n المتلقّاة في العمليّة	mmm (q, n) == true	القيمة المُعادة

(2) اشرحوا ماذا تنفّذ العمليّة what .

(3) ما هي تعقيدات زمن تشغيل العمليّة what ؟ علِّلوا إجابتكم.

/ يتبع في صفحة 6 /

للّذين يحلُّون بلغة #C

أ. أمامكم العمليّة Mmm ، التي تتلقّى دَوْرًا – q فيه أعداد أكبر من 0 ، وعددًا صحيحًا – z أكبر من 0 .

```
public static bool Mmm (Queue<int> q, int z)
{
    q.Insert (0);
    int num = q.Head();
    int y = 0;
    while (q.Head() > 0)
    {
        if (y < z)
        {
            if (q.Head() == num)
            {
                y++;
            }
            else
            {
                num = q.Head();
                y = 1;
            }
        }
        q.Insert (q.Remove());
    }
    q.Remove();
    return y == z;
}
```

معطى الدَّوْر q من نمط صحيح:

رأس الدَّوْر							نهاية الدَّوْر
	1	3	1	1	1	2	

(1) تتبَّعوا، بواسطة جدول المتابعة الذي أمامكم العمليّة Mmm(q , 4) ، واكتبوا ماذا تُعيد العمليّة.

الدَّوْر q	num	y	y < z	q.Head() == num

(2) اشرحوا ماذا تنفّذ العمليّة Mmm .

(3) ما هي تعقيدات زمن تشغيل العمليّة Mmm ؟ علِّلوا إجابتكم.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

ب. أمامكم العمليّة What ، التي تتلقّى دَوْرًا – q فيه أعداد أكبر من 0 ، وكِبَر الدَّوْر – n .

```
public static int What (Queue<int> q, int n)
{
    if (Mmm (q, n))
        return n;
    return What (q, n − 1);
}
```

معطى الدَّوْر – q من نمط صحيح:

رأس الدَّوْر							نهاية الدَّوْر
	1	3	1	1	1	2	

(1) تتبَّعوا العمليّة What (q, 6) ، واكتبوا ماذا تُعيد العمليّة (لا حاجة لمتابعة العمليّة Mmm) .
تحوي المتابعة، في كلّ استدعاء، قيمتَيْ q ، n والقيمة المُعادة.
أمامكم اقتراح لجدول متابعة (استعمال هذا الجدول ليس إلزاميًّا) .

الدَّوْر q المتلقَّى في العمليّة	قيمة n المتلقّاة في العمليّة	Mmm (q, n) == true	القيمة المُعادة

(2) اشرحوا ماذا تنفّذ العمليّة What.

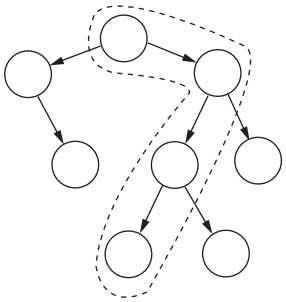
(3) ما هي تعقيدات زمن تشغيل العمليّة What ؟ علِّلوا إجابتكم.

/ يتبع في صفحة 8 /

3. "شجرة K – يسرى" هي شجرة بينارية، كلّ مسار فيها من جذر الشجرة إلى ورقة ما، يحوي على الأكثر K أقواس تتّجه يسارًا (أي على الأكثر K أبناء أياسر (שמאליים)).

انتبهوا: في "شجرة K – يسرى"، يمكن ألّا يكون أيّ مسار يحوي K أقواس تتّجه يسارًا (أي يمكن أن يكون في كلّ واحد من المسارات عدد أقواس يسرى أقلّ من K).

مثال:



هذه الشجرة ليست "شجرة 1 – يسرى"، لأنّه يوجد في الشجرة مسار فيه أكثر من قوس واحد أيسر (في المسار المخطَّط يوجد قوسان أيسران).

من جهة أخرى، هذه الشجرة هي "شجرة 2 – يسرى"، لأنّه لا يوجد فيها مسار فيه أكثر من قوسين أيسرَيْن (وبنفس المدى، هي أيضًا "شجرة 3 – يسرى"، وأيضًا "شجرة 4 – يسرى"، وهكذا).

أ. ارسموا "شجرة 1 – يسرى" (أي أنّه في كلّ مسار للشجرة، يوجد على الأكثر قوس واحد أيسر)، فيها 8 عُقَد وكَحَدٍّ أقصى 4 مستويات.

ب. اكتبوا عمليّة خارجيّة باسم isLeftK بلغة Java أو IsLeftK بلغة C#، تتلقّى شجرة بينارية – root من نمط صحيح ليست فارغة، وقيمة K ليست سالبة، وتُعيد true إذا كانت الشجرة هي "شجرة K – يسرى"، وإذا لم تكن كذلك – تُعيد العمليّة false.

/ يتبع في صفحة 9 /

الفصل الثاني (50 درجة)

في هذا الفصل أسئلة في ثلاثة مسارات:
ألغوريثمات، في الصفحات 9-12.
موديلات حسابيّة، في الصفحات 13-15.
برمجة موجّهة كائنات (بلغة Java وبلغة C#)، في الصفحات 16-27.
يجب الإجابة عن سؤالين في المسار الذي تعلّمتموه. (لكلّ سؤال – 25 درجة.)

ألغوريثمات

4. في هذ السؤال بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. أمامكم ستّة ادّعاءات 1-6. اختاروا أربعة منها، اكتبوا رقم الادّعاء، واذكروا بالنسبة لكلّ ادّعاء اخترتموه إذا كان صحيحًا أم غير صحيح. إذا كان الادّعاء صحيحًا – علِّلوا لماذا، وإذا كان غير صحيح – أعطوا مثالًا مناقضًا من رسم بيانيّ فيه 4 رؤوس على الأقلّ.

(1) معطى رسم بيانيّ ليس موجَّهًا، فيه n رؤوس. إذا كان في الرسم البيانيّ $n-1$ أقواس، فبالضرورة ليس فيه دوائر.

(2) معطى رسم بيانيّ موزون (ليس سالبًا) فيه دائرة واحدة على الأقلّ. الشجرة الامتداديّة الصغرى للرسم البيانيّ بالضرورة لا تحوي القوس الذي وزنه هو الأكبر في الرسم البيانيّ.

(3) معطى رسم بيانيّ ليس موجَّهًا فيه n رؤوس ومركِّب ارتباط واحد. يمكن أنّه، بعد مَحْو رأس واحد (والأقواس التي ترتبط به)، سيكون في الرسم البيانيّ $n-1$ مركِّبات ارتباط.

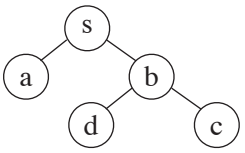
(4) معطى رسم بيانيّ ليس موجَّهًا بدون دوائر. مَسْحُ DFS يجد دائمًا أصغر بُعد ممكن بين رأسَيْن في الرسم البيانيّ يوجد بينهما مسار.

(5) معطى رسم بيانيّ موجَّه بدون دوائر. مَسْحُ DFS يجد دائمًا أصغر بُعد ممكن بين رأسَيْن يوجد بينهما مسار.

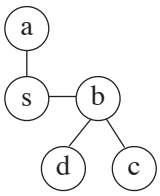
(6) معطى رسم بيانيّ ليس موجَّهًا. في الشجرة الامتداديّة DFS للرسم البيانيّ التي شُغِّلَت من رأس البداية s، يوجد رأس **v** له x أبناء. لذلك في كلّ شجرة امتداديّة DFS للرسم البيانيّ شُغِّلَت من رأس البداية **v**، يوجد للرأس **v** بالضرورة x أبناء على الأقلّ.

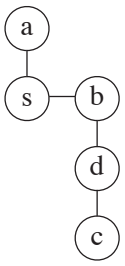
(انتبهوا: تكملة السؤال في الصفحة التالية.)

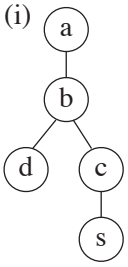
ب. معطى رسم بيانيّ ليس موجَّهًا G فيه 5 رؤوس: a، b، c، d، s .
بعد تشغيل DFS من رأس البداية **s** نتجت الشجرة التي في الرسم المعطى:



بالنسبة لكلّ واحدة من الشجرات الثلاث التي أمامكم، حدِّدوا هل من الممكن أنّها نتجت من تشغيل DFS على نفس الرسم البيانيّ G ، من رأس البداية **a** . علِّلوا تحديداتكم.

(iii)


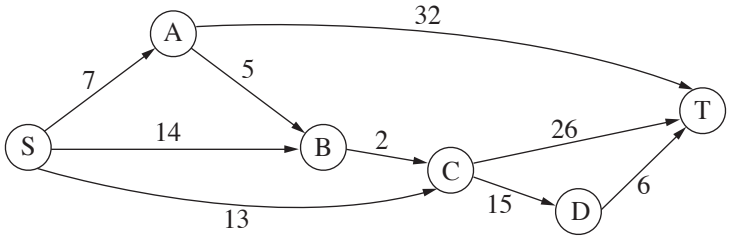
(ii)


(i)


/ يتبع في صفحة 11 /

5. في هذ السؤال بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. أمامكم رسم بيانيّ موزون G_1 :

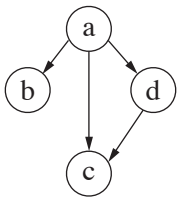


(1) أيّ ألغوريثم يُعيد المسار الأقصر (الأقلّ وَزْنًا) بين رأسَيْن في رسم بيانيّ موزون (ليس سالبًا) أيًّا كان؟
ما هي تعقيدات زمن تشغيل الألغوريثم؟ علِّلوا إجابتكم.
إضافةً إلى ذلك، شغِّلوا الألغوريثم الذي كتبتموه، بالنسبة للرسم البيانيّ المعطى G_1 ، من العقدة S إلى العقدة T .
نَفِّذوا متابعة، مرحلة تلو الأخرى، واكتبوا المسار وطوله (مجموع أوزانه).

(2) بالنسبة لرسم بيانيّ G موزون أيًّا كان، شَغِّلْنا الألغوريثم الذي يجد المسار الأقصر (الأقلّ وَزْنًا) بين رأسَيْن، وحصلنا على مسار يمرّ عبر الأقواس $E(e_i , e_j , e_m , ...)$.

- هل المسار الأقصر بين الرأسَيْن سيبقى بالضرورة مطابقًا (سيبقى نفسه) إذا ضُرِب في x $(1 < x)$ وَزْنُ كلّ واحد من الأقواس في الرسم البيانيّ؟ علِّلوا إجابتكم.
- هل المسار الأقصر بين الرأسَيْن سيبقى بالضرورة مطابقًا (سيبقى نفسه) إذا كَبُر بـ x $(1 < x)$ وَزْنُ كلّ واحد من الأقواس في الرسم البيانيّ؟ علِّلوا إجابتكم.

ب. أمامكم رسم بيانيّ موجَّه G :



(1) اعرضوا مَسْحَيْ DFS من رأس البداية **a** ، بحيث تَنْتُج شجرتان ارتفاعاهما مختلفان.

(2) أضيفوا أصغر عدد ممكن من الأقواس، بحيث تَنْتُج في ثلاثة مُسُوح DFS من الرأس **a** ، ثلاث شجرات ارتفاعاتها مختلفة. ارسموا الرسم البيانيّ بعد الإضافة، واعرضوا ثلاثة المُسُوح.

/ يتبع في صفحة 12 /

6. رسم بيانيّ ليس موجَّهًا يُسمّى "**رسمًا بيانيًّا – تقريبًا – شجرة**"، إذا كان فيه قوس عندما نُزيله نحصل على شجرة.

أ. ارسموا رسمًا بيانيًّا فيه 6 رؤوس وهو "**رسم بيانيّ – تقريبًا – شجرة**".

ب. معطى "**رسم بيانيّ – تقريبًا – شجرة**"، فيه n رؤوس.

(1) هل يمكن معرفة عدد الأقواس في هذا الرسم البيانيّ؟ علِّلوا إجابتكم. إذا أجبتم أنّه يمكن، اذكروا عدد الأقواس.

(2) هل يمكن معرفة عدد مركِّبات الارتباط في هذا الرسم البيانيّ؟ علِّلوا إجابتكم. إذا أجبتم أنّه يمكن، اذكروا عدد مركِّبات الارتباط.

جـ. معطى رسم بيانيّ ليس موجَّهًا G فيه n رؤوس، ممثَّل بواسطة قائمة مجاوَرات.

(1) اكتبوا ألغوريثمًا ناجعًا يُعيد "صدق" إذا كان الرسم البيانيّ هو "**رسم بيانيّ – تقريبًا – شجرة**"، خلاف ذلك يُعيد "كذب".

(2) ما هي تعقيدات زمن تشغيل الألغوريثم الذي كتبتموه؟ علِّلوا إجابتكم.

/ يتبع في صفحة 13 /

موديلات حسابيّة

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

أ. أمامكم ثلاث لغات فوق الأبجديّة {a , b}:

$L_1 = \{ a^n b^m a^k \mid n, m, k > 0, n = k \}$

$L_2 = \{ a^n b^m a^k \mid n, m, k > 0, n \% 2 = k \% 2 \}$

$L_3 = \{ a^n b^m a^k \mid n, m, k > 0, n+m+k < 100 \}$

أمامكم أربعة بنود (1) – (4). أجيبوا عن جميعها (في التعليل، يمكن الاكتفاء بشرح كلاميّ، لا حاجة إلى أوتومات).

(1) بالنسبة لكلّ واحدة من اللغات $L_3 - L_1$، اكتبوا إذا كانت نظاميّة أم غير نظاميّة. علّلوا إجابتكم.

(2) هل اللُّغة $L_1 \cap L_3$ نظاميّة؟ علّلوا إجابتكم.

(3) هل اللُّغة $L_1 \cap \overline{L_3}$ نظاميّة؟ علّلوا إجابتكم.

(4) هل اللُّغة $L_2 \cap \overline{L_3}$ نظاميّة؟ علّلوا إجابتكم.

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

$L = \{a^m b^k c^x \mid m, k, x > 0$ **وأيضًا** $(m\%2 = k\%2$ **أو** $m\%2 = x\%2)\}$

الشرح: اللُّغة L تحوي كلمات يظهر فيها كلّ واحد من الحروف abc مرّةً واحدةً على الأقلّ، وحسب هذا الترتيب (ظهورات a في البداية وبعد ذلك ظهورات b وبعد ذلك ظهورات c). كذلك، إذا كان عدد مرّات ظهور الحرف a زوجيًّا/ فرديًّا، فإنّ عدد مرّات ظهور الحرف b أو الحرف c سيكون هو أيضًا زوجيًّا/ فرديًّا بالتلاؤم.

مثال لكلمة في اللُّغة L هي abbbbccc، لأنّ كلّ واحد من الحروف abc يظهر فيها، وحسب الترتيب المطلوب. كذلك، عدد مرّات ظهور الحرف a هو فرديّ، وهكذا أيضًا عدد مرّات ظهور الحرف c .

مثال إضافيّ لكلمة في اللُّغة L هي aaaabbc، لأنّ كلّ واحد من الحروف abc يظهر فيها، وحسب الترتيب المطلوب. كذلك، عدد مرّات ظهور الحرف a هو زوجيّ، وهكذا أيضًا عدد مرّات ظهور الحرف b .

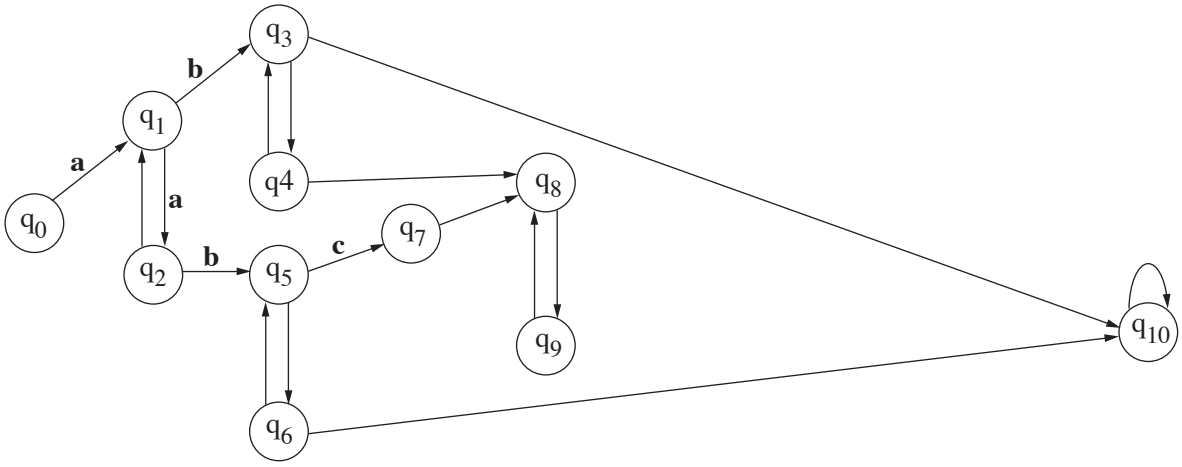
مثال لكلمة **ليست** في اللُّغة L هي aabc. رغم أنّ كلّ واحد من الحروف abc يظهر فيها، وحسب الترتيب المطلوب، إلّا أنّ عدد مرّات ظهور الحرف a هو زوجيّ، بينما عدد مرّات ظهور الحرف b والحرف c هو فرديّ.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 14 /

(1) أمامكم خمس كلمات: abbcc، abcc، bbc، abc، abbac.
انسخوا كلّ واحدة من الكلمات إلى دفتركم، وحدِّدوا إذا كانت تتبع للُّغة L أم لا تتبع للُّغة L. علِّلوا تحديداتكم.

(2) معطى أمامكم أوتومات نهائيّ محدود ليس كاملًا يتلقّى اللُّغة L. توجد في الأوتومات جميع الحالات وجميع الانتقالات اللازمة، لكن في بعض الانتقالات، تنقص إشارات الإدخال (الحروف في الانتقالات)، والحالات المتلقّية في الأوتومات غير مُشار إليها بتاتًا.
انسخوا الأوتومات إلى دفتركم، أضيفوا إشارات الإدخال الناقصة، وأشيروا إلى الحالات المتلقّية.
ملاحظة: يجب أن يبقى الأوتومات نهائيًّا محدودًا ليس كاملًا. لا تضيفوا إليه حالات أو انتقالات، ولا تُغيِّروا إشارات الإدخال التي تظهر فيه. الإجابة التي تتضمّن تغييرًا في الأوتومات المعطى لن تحصل على درجات.



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

أ. معطاة ثلاث لغات L_1 ، L_2 ، L_3 فوق الأبجديّة $\{a, b, c\}$.
أمامكم ثلاثة ادّعاءات. بالنسبة لكلّ واحد من الادّعاءات، حدِّدوا إذا كان صحيحًا أم غير صحيح.
إذا كان الادّعاء صحيحًا – اشرحوا لماذا، وإذا كان غير صحيح – ادحضوه من خلال مثال مناقض: اختاروا لغات معيّنة، وبيِّنوا أنّ الادّعاء لا يتحقّق بالنسبة لها.

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

(2) $L_1 \cdot (L_2 \cup L_3) = (L_1 \cdot L_2) \cup (L_1 \cdot L_3)$

(3) $L_1 \cap (L_2 \cdot L_3) = (L_1 \cap L_2) \cdot (L_1 \cap L_3)$

ب. معطاة اللُّغة L فوق الأبجديّة $\{a, b, c\}$:
ابنوا أوتومات راصّة محدودًا يتلقّى اللُّغة L.

$L = \{a^i b^j c^{i-j} \mid i \geq 4, (i-j) \geq 3\}$

/ يتبع في صفحة 15 /

9. اكتبوا آلة تيورينج تتلقّى في بداية الشريط عددًا أونارِيًّا أكبر من واحد، وتُعيد عددًا جديدًا كما هو مفصَّل أمامكم:

- إذا كان العدد المتلقَّى زوجيًّا – تُعيد الآلة العدد الأصليّ مقسومًا على 2 .
- إذا كان العدد المتلقَّى فرديًّا – تُعيد الآلة العدد الأصليّ ناقص 1 .

توجيهات:

- لا حاجة للحفاظ على المُدخَلات (يمكن تغيير المُدخَل إلى إشارات مختلفة) .
- العدد المُعاد يظهر في مكان ما في الشريط بين إشارتَي $.
- لا حاجة لفحص سلامة المُدخَلات.

مثال لعدد زوجيّ:

الشريط قبل التشغيل:

	–	1	1	1	1	1	1	△	△	△	△	△

الشريط بعد التشغيل:

...	$	1	1	1	$

مثال لعدد فرديّ:

الشريط قبل التشغيل:

	–	1	1	1	1	1	△	△	△	△	△

الشريط بعد التشغيل:

...	$	1	1	1	1	$

/ يتبع في صفحة 16 /

برمجة موجّهة كائنات

انتبهوا: الأسئلة 10–12 هي للّذين يحلُّون بلغة Java وكذلك للّذين يحلُّون بلغة #C .
لكلّ سؤال صيغة بلغة Java وصيغة بلغة #C .

10. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 16–17، وبلغة **#C** في الصفحتين 18–19.

للّذين يحلُّون بلغة Java

في حديقة الحيوانات "نعيش بسلام"، طُوِّرَت منظومة مُحَوْسَبة يحفظون فيها معطيات عن الحيوانات التي في الحديقة.
تحوي المنظومة الفئات التالية: حيوان – **Animal** ، عصفور – **Bird** ، ببّغاء – **Parrot** ، سمكة – **Fish** ، أفعى – **Snake** .
معطى قسم من مبنى الفئات في المنظومة المُحَوْسَبة:

```
public class Animal {
    private String color;
    private int weight;
    public Animal (String color, int weight)
    {
        this.color = color;
        this.weight = weight;
    }
    public String toString() {
        return "My color is " + this.color + "! And I weigh " + weight + " kilos!";
    }
}
public class Bird {
    private String color;
    private int weight;
    private boolean isFlying;
    public void zoo () {
        System.out.println ("Hello");
    }
}
public class Parrot {
    private String color;
    private int weight;
    private boolean isFlying;
    private boolean isTalking;
    public void zoo () {
        System.out.println ("Hello");
    }
}
public class Fish {
    private String color;
    private int weight;
    private String waterType;     // سمكة بحر/ سمكة مياه عذبة
}
public class Snake {
    private String color;
    private int weight;
    private int length;
    private boolean isVenomous;     // هل الأفعى سامّة؟
}
```

افترِضوا أنّه توجد عمليّات get وَ set لصفات الفئة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 17 /

أ. اكتبوا من جديد عناوين وصفات وعمليّات الفئات: **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، حسب مبادئ البرمجة الموجّهة كائنات (لا تُغيِّروا الفئة Animal) . في هذا البند، لا حاجة لكتابة عمليّات بنائيّة . افترِضوا أنّ العمليّتَيْن get وَ set موجودتان . إضافةً إلى ذلك، ارسموا مخطَّطًا هرميًّا بين جميع الفئات . يجب الإشارة إلى توريث بواسطة السهم ◁—— .

ب. معطاة الفئة Tester :

```
public class Tester {
    public static void main (String[] args) {
        Animal[] animals = new Animal [5];
        animals [0] = new Bird ("white", 4, false);
        animals [1] = new Fish ("blue", 3, "sweet water");
        animals [2] = new Parrot ("brown", 12, true, true);
        animals [3] = new Snake ("gray", 2, 6, true);
        animals [4] = new Snake ("black", 3, 4, false);
        //***
    }
}
```

أضيفوا عمليّات بنائيّة إلى الفئات **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، تبتدئ صفات الفئات (بحيث تعمل العمليّة main بدون أخطاء) .

ملاحظة: لا تُغيِّروا العمليّة البنائيّة المعطاة لِـ **Animal** .

جـ. (1) في العمليّة main ، أُضيفَت قطعة الكود التي أمامكم (في المكان المشار إليه بِـ ***) :

```
for (int i = 0; i < animals.length; i++) {
    System.out.println (animals[i]); }
```

قطعة الكود سليمة، وتطبع المُخرَج الذي أمامكم:

```
I'm a bird! My color is white! And I weigh 4 kilos!
My color is blue! And I weigh 3 kilos!
I'm a parrot! My color is brown!
I'm a snake! I'm venomous, be careful! My color is gray! And I weigh 2 kilos!
I'm a snake! I'm not venomous! My color is black! And I weigh 3 kilos!
```

أضيفوا العمليّة toString إلى الفئات (فقط أين تدعو الحاجة) حسب مبادئ البرمجة الموجّهة كائنات، بحيث يكون المُخرَج ملائمًا .

ملاحظة: لا تُغيِّروا العمليّة toString في الفئة **Animal** . الإجابة التي تتضمّن تغييرًا لن تحصل على درجات .

(2) في الفئة Tester ، أُضيفَت العمليّة التي أمامكم:

```
public static void hello (Animal [] arr) {
    for (int i = 0; i < arr.length; i++) {
        arr[i].zoo(); }
}
```

إذا كانت العمليّة سليمة، اكتبوا مُخرَجها بالنسبة للمصفوفة animals (المعطاة أعلاه)، وإذا لم تكن العمليّة سليمة – صحِّحوها (بحيث يكون استدعاء zoo داخل الحلقة سليمًا)، واكتبوا المُخرَج بالنسبة للمصفوفة animals .

ملاحظتان: – لا تُغيِّروا الفئات نفسها . الإجابة التي تتضمّن تغييرًا في الفئات لن تحصل على درجات .

– يجب أن تعمل العمليّة بدون أخطاء بالنسبة لكلّ مصفوفة من نمط Animal .

/ يتبع في صفحة 18 /

للّذين يحلُّون بلغة #C

في حديقة الحيوانات "نعيش بسلام"، طُوِّرَت منظومة مُحَوْسَبة يحفظون فيها معطيات عن الحيوانات التي في الحديقة.

تحوي المنظومة الفئات التالية: حيوان – **Animal**، عصفور – **Bird**، ببّغاء – **Parrot**، سمكة – **Fish**، أفعى – **Snake**.

معطى قسم من مبنى الفئات في المنظومة المُحَوْسَبة:

```
public class Animal {
    private string color;
    private int weight;
    public Animal (string color, int weight)
    {
        this.color = color;
        this.weight = weight;
    }
    public override string ToString() {
        return "My color is " + this.color + "! And I weigh " + weight + " kilos!";
    }
}
public class Bird {
    private string color;
    private int weight;
    private bool isFlying;
    public void Zoo () {
        Console.WriteLine ("Hello");
    }
}
public class Parrot {
    private string color;
    private int weight;
    private bool isFlying;
    private bool isTalking;
    public void Zoo () {
        Console.WriteLine ("Hello");
    }
}
public class Fish {
    private string color;
    private int weight;
    private string waterType;     // سمكة بحر/ سمكة مياه عذبة
}
public class Snake {
    private string color;
    private int weight;
    private int length;
    private bool isVenomous;     // هل الأفعى سامّة؟
}
```

افترِضوا أنّه توجد عمليّات Get وَ Set لصفات الفئة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 19 /

أ. اكتبوا من جديد عناوين وصفات وعمليّات الفئات: **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، حسب مبادئ البرمجة الموجّهة كائنات (لا تُغيِّروا الفئة Animal) . في هذا البند، لا حاجة لكتابة عمليّات بنائيّة . افترِضوا أنّ العمليّتَيْن Get وَ Set موجودتان . إضافةً إلى ذلك، ارسموا مخطَّطًا هرميًّا بين جميع الفئات . يجب الإشارة إلى توريث بواسطة السهم ——◁ .

ب. معطاة الفئة Tester :

```
public class Tester {
    public static void Main (string[] args) {
        Animal[] animals = new Animal [5];
        animals [0] = new Bird ("white", 4, false);
        animals [1] = new Fish ("blue", 3, "sweet water");
        animals [2] = new Parrot ("brown", 12, true, true);
        animals [3] = new Snake ("gray", 2, 6, true);
        animals [4] = new Snake ("black", 3, 4, false);
        //***
    }
}
```

أضيفوا عمليّات بنائيّة إلى الفئات **Bird** وَ **Parrot** وَ **Fish** وَ **Snake** ، تبتدئ صفات الفئات (بحيث تعمل العمليّة Main بدون أخطاء) .

ملاحظة: لا تُغيِّروا العمليّة البنائيّة المعطاة لِـ **Animal** .

جـ. (1) في العمليّة Main ، أُضيفَت قطعة الكود التي أمامكم (في المكان المشار إليه بِـ ***) :

```
for (int i = 0; i < animals.Length; i++) {
    Console.WriteLine (animals[i]); }
```

قطعة الكود سليمة، وتطبع المُخرَج الذي أمامكم:

```
I'm a bird! My color is white! And I weigh 4 kilos!
My color is blue! And I weigh 3 kilos!
I'm a parrot! My color is brown!
I'm a snake! I'm venomous, be careful! My color is gray! And I weigh 2 kilos!
I'm a snake! I'm not venomous! My color is black! And I weigh 3 kilos!
```

أضيفوا العمليّة ToString إلى الفئات (فقط أين تدعو الحاجة) حسب مبادئ البرمجة الموجّهة كائنات، بحيث يكون المُخرَج ملائمًا .

ملاحظة: لا تُغيِّروا العمليّة ToString في الفئة **Animal** . الإجابة التي تتضمّن تغييرًا لن تحصل على درجات .

(2) في الفئة Tester ، أُضيفَت العمليّة التي أمامكم:

```
public static void Hello (Animal [] arr) {
    for (int i = 0; i < arr.Length; i++) {
        arr[i].Zoo(); }
}
```

إذا كانت العمليّة سليمة، اكتبوا مُخرَجها بالنسبة للمصفوفة animals (المعطاة أعلاه)، وإذا لم تكن العمليّة سليمة – صحِّحوها (بحيث يكون استدعاء Zoo داخل الحلقة سليمًا)، واكتبوا المُخرَج بالنسبة للمصفوفة animals .

ملاحظتان: – لا تُغيِّروا الفئات نفسها . الإجابة التي تتضمّن تغييرًا في الفئات لن تحصل على درجات .

– يجب أن تعمل العمليّة بدون أخطاء بالنسبة لكلّ مصفوفة من نمط Animal .

/ يتبع في صفحة 20 /

11. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 20–21، وبلغة **C#** في الصفحتين 22–23.

للّذين يحلُّون بلغة Java

معطاة الفئات One ، Two ، Three :

```
public class One {
	public static int count = 0;
	private int number;
	public One() {
		count++;
		number = count;
	}
	public One (int num) {
		number = num;
	}
	public String toString() {
		return count + ",  "+ number;
	}
}
-----------------------------------------------
public class Two extends One {
	private String strTwo;
	public Two() {
		strTwo = "Fast";
	}
	public Two (String s) {
		super (15);
		strTwo = s;
	}
	public String toString() {
		return super.toString() + "  "+strTwo;
	}
}
-----------------------------------------------
public class Three extends Two {
	private String strThree;
	public Three (String s) {
		strThree = s;
	}
	public String toString() {
		return super.toString() + "  "+ strThree;
	}
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/يتبع في صفحة 21/

أ. معطاة الفئة Test :

```
public class Test
{
    public static void main (String[] args)
    {
        One[] arr=new One[6];
        arr[0]= … // (1)
        arr[1]= … // (2)
        arr[2]= … // (3)
        arr[3]= … // (4)
        arr[4]= … // (5)
        arr[5]= … // (6)
        for (int i = 0; i < arr.length; i++) {
            System.out.println (arr[i]);
        }
    }
}
```

أكمِلوا الأسطر (1)–(6)، بحيث يكون المُخرَج:

```
4, 1 Fast
4, 2 Fast Car
4, 3 Fast Horse
4, 15 Small
4, 15 Tall
4, 4
```

ب. عنوان الفئة Three تَغيَّرَ إلى العنوان:

public class Three extends One

هل يؤدّي هذا التغيير إلى خطأ؟ إذا كان يؤدّي – اشرحوا الخطأ، إذا لم يكن يؤدّي – اكتبوا ماذا سيكون مُخرَج العمليّة main بعد التغيير (الأسطر الناقصة هي كما كتبتموها في البند "أ").

ملاحظة: ليس هناك تغييرات إضافيّة في الفئة.

/ يتبع في صفحة 22 /

للّذين يحلُّون بلغة #C

معطاة الفئات Three , Two , One :

```
public class One {
	public static int count = 0;
	private int number;
	public One() {
		count++;
		number = count;
	}
	public One (int num) {
		number = num;
	}
	public override string ToString() {
		return count + ",  "+ number;
	}
}
-------------------------------------------
public class Two : One {
	private string strTwo;
	public Two() {
		strTwo = "Fast";
	}
	public Two (string s) : base (15) {
		strTwo = s;
	}
	public override string ToString() {
		return base.ToString() + "  "+strTwo;
	}
}
-------------------------------------------------
public class Three : Two {
	private string strThree;
	public Three (string s) {
		strThree = s;
	}
	public override String ToString() {
		return base.ToString() + "  "+ strThree;
	}
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 23 /

أ. معطاة الفئة Test :

```
public class Test
{
    public static void Main (string[] args)
    {
        One[] arr=new One[6];
        arr[0]= … // (1)
        arr[1]= … // (2)
        arr[2]= … // (3)
        arr[3]= … // (4)
        arr[4]= … // (5)
        arr[5]= … // (6)
        for (int i = 0; i < arr.Length; i++) {
            Console.WriteLine (arr[i]);
        }
    }
}
```

أكمِلوا الأسطر (1)–(6)، بحيث يكون المُخرَج:

```
4, 1 Fast
4, 2 Fast Car
4, 3 Fast Horse
4, 15 Small
4, 15 Tall
4, 4
```

ב. عنوان الفئة Three تَغيَّرَ إلى العنوان:

public class Three : One

هل يؤدّي هذا التغيير إلى خطأ؟ إذا كان يؤدّي – اشرحوا الخطأ، إذا لم يكن يؤدّي – اكتبوا ماذا سيكون مُخرَج العمليّة Main بعد التغيير (الأسطر الناقصة هي كما كتبتموها في البند "أ").

ملاحظة: ليس هناك تغييرات إضافيّة في الفئة.

/ يتبع في صفحة 24 /

12. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 24–25، وبلغة **C#** في الصفحتين 26–27.

<u>للّذين يحلُّون بلغة Java</u>

معطاة الفئات **A** ، **B** ، **C** ، **D** :

```
public class A {
    protected int i;
    public A (int i) { this.i = i; }
}
```

```
public class B extends A {
    public B (int i) { super (i + 1); }
    public  boolean foo (Object myObject)  {
        System.out.println ("foo1");
        return ((myObject instanceof B) && (this.i == ((B)myObject).i));
    }
    public boolean foo (B myB, int num)  {
        System.out.println ("foo2");
        return (this.i +  myB.i)  <  num;
    }
}
```

```
public class C extends B {
    public C (int i)  { super (i); }
    public boolean foo (int num)  {
        System.out.println ("foo3");
        return (this.i != num);
    }
}
```

```
public class D extends B {
    public D (int i)  { super (i * 2); }
    public boolean foo (B myB, int num)    {
        System.out.println ("foo4");
        return super.foo (myB, 20);
    }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 25 /

أ. ارسموا المخطّط الهرميّ للفئات A ، B ، C ، D . يجب الإشارة إلى توريث بواسطة السهم ⟵ .

ب. معطاة الفئة Tester :

```
public class Tester {
    public static void main (String[] args) {
        A a = new A (1);
        B b = new B (2);
        C c = new C (3);
        D d = new D (4);
        B bd = new D (5);
        A ac = new C (6);
        //***
    }
}
```

(1) ارسموا الكائنات التي تكوّنت في العمليّة main .

(2) في المكان المُشار إليه بِـ ***، أُضيفَت قطعة الكود السليم التي أمامكم:

```
System.out.println (c.foo(5));          //(1)
System.out.println (d.foo(a));          //(2)
System.out.println (bd.foo(b));         //(3)
System.out.println (bd.foo(b, 1));      //(4)
System.out.println (((C)ac).foo(c));    //(5)
```

اكتبوا مُخرَجات العمليّة (اكتبوا رقم السطر التابع لكلّ مُخرَج).

/ يتبع في صفحة 26 /

للّذين يحلُّون بلغة #C

معطاة الفئات **A** ، **B** ، **C** ، **D** :

```
public class A {
      protected int i;
      public A (int i) { this.i = i; }
}
-----------------------------------------------------------------------------------------------------
public class B : A {
      public B (int i) : base (i + 1)   { }
      public  bool Foo (Object myObject)  {
            Console.WriteLine ("Foo1");
            return ((myObject is B) && (this.i == ((B)myObject).i));
      }
      public virtual bool Foo (B myB, int num)  {
            Console.WriteLine ("Foo2");
            return (this.i +  myB.i)  <  num;
      }
}
-----------------------------------------------------------------------------------------------------
public class C : B {
      public C (int i) : base (i) { }
      public bool Foo (int num)  {
            Console.WriteLine ("Foo3");
            return (this.i != num);
      }
}
-----------------------------------------------------------------------------------------------------
public class D : B {
      public D (int i) : base (i * 2) { }
      public override bool Foo (B myB, int num)    {
            Console.WriteLine ("Foo4");
            return base.Foo (myB, 20 );
      }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 27 /

أ. ارسموا المخطّط الهرميّ للفئات A ، B ، C ، D . يجب الإشارة إلى توريث بواسطة السهم ◁—— .

ب. معطاة الفئة Tester :

```
public class Tester {
    public static void Main (string[] args) {
        A a = new A (1);
        B b = new B (2);
        C c = new C (3);
        D d = new D (4);
        B bd = new D (5);
        A ac = new C (6);
        //***
    }
}
```

(1) ارسموا الكائنات التي تكوّنت في العمليّة Main .

(2) في المكان المُشار إليه بِـ ***، أُضيفَت قطعة الكود السليم التي أمامكم:

```
Console.WriteLine (c.Foo(5));          //(1)
Console.WriteLine (d.Foo(a));          //(2)
Console.WriteLine (bd.Foo(b));         //(3)
Console.WriteLine (bd.Foo(b, 1));      //(4)
Console.WriteLine (((C)ac).Foo(c));    //(5)
```

اكتبوا مُخرَجات العمليّة (اكتبوا رقم السطر التابع لكلّ مُخرَج) .

בהצלחה!
نتمنّى لكم النّجاح!