מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ו, 2026
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 2026
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 الّتي 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 **Order** – طلبيّة لزبون، لها صفتان:

- id – رقم هويّة الزبون، من نمط صحيح.
- count – كمّيّة المنتَجات التي طُلبت، من نمط صحيح.

افترِضوا أنّه توجد عمليّات get/Get وَ set/Set لصفتَي الفئة، وعمليّة بنائيّة تتلقّى قيَمًا بالنسبة لصفتَي الفئة.

في شركة الشحن "برق" بُنِيَ دَوْر qOrder – من نمط **Order**، يحفظ الطلبيّات المختلفة للزبائن في يوم معيّن، التي يجب على شركة الشحن أن تزوّدها.

ملاحظتان:

– يمكن أن يكون في الدَّوْر عدّة طلبيّات لنفس الزبون – id (إذا كان للزبون أكثر من طلبيّة واحدة في نفس اليوم).
– مكان الطلبيّات في الدَّوْر هو ليس حسب ترتيب معيّن (كذلك طلبيّات نفس الزبون يمكن أن تظهر في أماكن مختلفة في الدَّوْر).

في نهاية اليوم، من أجل زيادة نجاعة الشحن، يُجرون، في دَوْر جديد، توحيدًا للطلبيّات حسب رقم هويّة الزبون (id)، بحيث تُحفَظ لكلّ زبون في الدَّوْر طلبيّة واحدة فقط، مع العدد الكلّيّ لجميع المنتَجات التي طلبها في جميع الطلبيّات (count). هكذا مثلًا، إذا كان في الدَّوْر qOrder 3 طلبيّات لنفس الزبون: طلبيّة لِـ 20 منتَجًا وطلبيّة لِـ 15 منتَجًا وطلبيّة لِـ 30 منتَجًا، بعد التوحيد، يظهر الزبون مرّة واحدة فقط في الدَّوْر الجديد – مع طلبيّة لِـ 65 منتَجًا.

أ. **(1)** طبِّقوا العمليّة التي أمامكم:

Java – public static Queue<Order> uniteOrders (Queue<Order> qOrder)

C# – public static Queue<Order> UniteOrders (Queue<Order> qOrder)

تتلقّى العمليّة دَوْرًا فيه طلبيّات – qOrder، يمكن أن يظهر فيه نفس الزبون أكثر من مرّة واحدة، وتُعيد دورًا جديدًا فيه توحيد الطلبيّات كما شُرِح أعلاه.

ملاحظات:

– لا توجد أهمّيّة لترتيب الطلبيّات بعد توحيد الطلبيّات.
– لا تستعمِلوا في هذا البند مصفوفة أو قائمة مرتبطة (أو أيّ مبنى بيانات آخر باستثناء الدَّوْر). الحلّ الذي يتضمّن استعمالها لن يحصل على درجات.
– يمكن تغيير الدَّوْر المتلقَّى.

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

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 3 /

ب. (1) "زبون عاديّ" هو الزبون الذي يطلب **بالمجمل** أقلّ من 10 منتَجات في يوم معيّن، وَ"زبون مفضَّل" هو الزبون الذي يطلب **بالمجمل** 10 منتَجات أو أكثر في يوم معيّن.

طبِّقوا العمليّة التي أمامكم:

Java – public static Queue<Integer> preferredClients (Queue<Order> qOrder)

C# – public static Queue<int> PreferredClients (Queue<Order> qOrder)

تتلقّى العمليّة دَوْر طلبيّات في يوم معيّن – qOrder من نمط **Order** (دَوْرًا "ليس موحَّدًا"، يمكن أن يظهر فيه نفس الزبون عدّة مرّات)، وتُعيد دَوْرًا من نمط **صحيح**. في بداية الدَّوْر المُعاد، تظهر أرقام هويّات (id) كلّ مَن هو "زبون مفضَّل" وبعدها تظهر أرقام هويّات (id) كلّ مَن هو "زبون عاديّ" (بدون أهمّيّة لترتيب الزبائن في كلّ مجموعة).

بإمكانكم استعمال العمليّة التي كتبتموها في البند "أ".

ملاحظات:

- كلّ زبون (id) يظهر مرّة واحدة فقط في الدَّوْر المُعاد.
- لا تستعمِلوا في هذا البند مصفوفة أو قائمة مرتبطة (أو أيّ مبنى بيانات آخر باستثناء الدَّوْر). الحلّ الذي يتضمّن استعمالها لن يحصل على درجات.
- يمكن تغيير الدَّوْر المتلقّى.

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

/ يتبع في صفحة 4 /

2. معطى توجيهان إلى سلسلتَي حلقات:

numHead – توجيه إلى سلسلة حلقات لأعداد من نمط صحيح.

charHead – توجيه إلى سلسلة حلقات من نمط رمز (char)، تُمثَّل فيها العمليّتان الحسابيّتان الجمع والطرح ('+' ، '−').

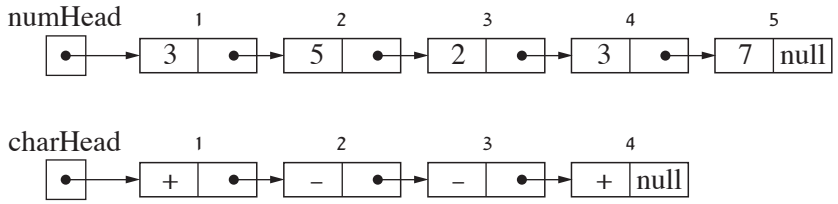
ملاحظة: كمّيّة الحلقات في سلسلة الأعداد هي أكبر بـ 1 من كمّيّة الحلقات في سلسلة الرموز.

الدمج بين السلسلتَيْن يُمثِّل تسلسل عمليّات حسابيّة، كما هو مفصَّل أمامكم:

الرمز في المكان 1 في سلسلة الرموز يصف العمليّة التي تأتي قبل العدد الذي في المكان 2 في سلسلة الأعداد، والرمز في المكان 2 يصف العمليّة التي تأتي قبل العدد الذي في المكان 3، والرمز في المكان 3 يصف العمليّة التي تأتي قبل العدد الذي في المكان 4، وهكذا حتّى نهاية السلسلتَيْن.

انتبهوا: لا توجد عمليّة حسابيّة قبل العدد الأوّل في سلسلة الأعداد.

مثال: السلسلتان اللتان أمامكم تُمثِّلان تسلسل العمليّات الحسابيّة: $7+3-2-5+3$.



أ. طبِّقوا العمليّة التي أمامكم:

Java – public static int eval (Node<Integer> numHead, Node<Character> charHead, int len)

C# – public static int Eval (Node<int> numHead, Node<char> charHead, int len)

len هو كمّيّة **الأعداد** التي نرغب في حسابها من داخل سلسلة الأعداد (لا توجد عمليّة حسابيّة قبل العدد الأوّل، لذلك كمّيّة الرموز هي len–1 بالتلاؤم).

تُعيد العمليّة نتيجة حساب len أعداد بتسلسل، بدءًا من بداية سلسلة الأعداد وبداية سلسلة الرموز.

افترِضوا أنّ len هو أكبر من 0 . لا تُغيِّروا السلسلتَيْن المتلقّاتَيْن.

ملاحظة: إذا كان len أكبر من كمّيّة الحلقات في سلسلة الأعداد، تُعيد العمليّة نتيجة حساب جميع الأعداد في السلسلة.

مثال: بالنسبة للسلسلتَيْن في المثال أعلاه وَ $\mathbf{len = 1}$ ، تُعيد العمليّة 3 ، لأنّه هو العدد الأوّل في سلسلة الأعداد (لِكَوْنه هو العدد الأوّل والوحيد، لا توجد أيّة عمليّة حسابيّة يجب تنفيذها).

مثال إضافيّ: بالنسبة لنفس السلسلتَيْن وَ $\mathbf{len = 3}$ ، تُعيد العمليّة 6 $(3+5-2=6)$.

مثال إضافيّ: بالنسبة لنفس السلسلتَيْن وَ $\mathbf{len = 8}$ ، تُعيد العمليّة 10 $(3+5-2-3+7=10)$.

الشرح: عدد الحلقات في سلسلة الأعداد هو أقلّ من 8 ، لذلك تُعيد العمليّة نتيجة حساب جميع الأعداد في السلسلة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

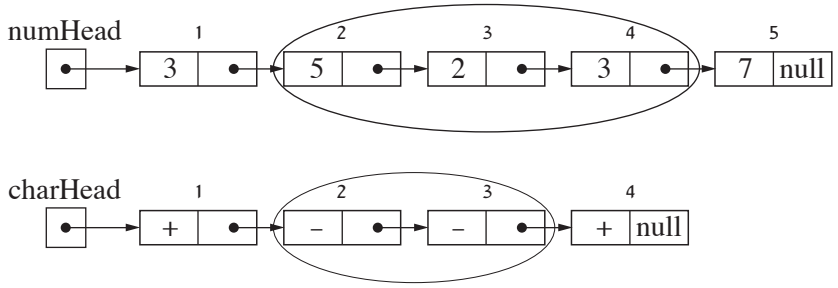
ب. طبِّقوا العمليّة التي أمامكم:

Java – public static boolean match (Node<Integer> numHead, Node<Character> charHead, int len, int val)

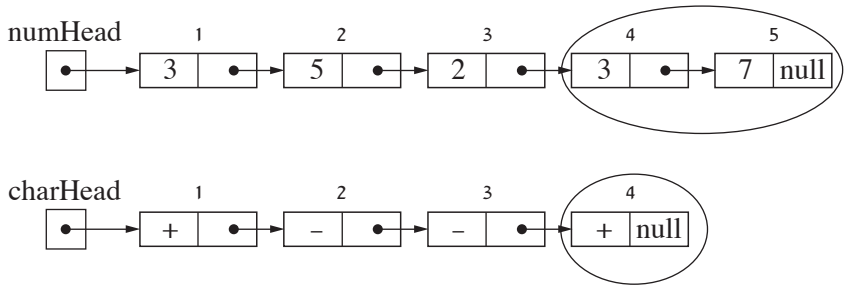
C# – public static bool Match (Node<int> numHead, Node<char> charHead, int len, int val)

تُعيد العمليّة true إذا كانت هناك حلقة **في مكان ما** في سلسلة الأعداد وحلقة **في مكان مطابق** في سلسلة الرموز، اللذين منهما فصاعدًا يوجد **len** أعداد بتسلسل نتيجةُ حسابِها هي **val** (أي أنّه إذا بدأنا الحساب **من الحلقة الثانية** في سلسلة الأعداد، ستكون العمليّة الحسابيّة الأولى أيضًا هي **الحلقة الثانية** في سلسلة الرموز). خلاف ذلك، تُعيد العمليّة false.
بإمكانكم استعمال العمليّة التي كتبتموها في البند "أ". افترِضوا أنّ len أكبر من 0. لا تُغيِّروا السلسلتَيْن المتلقّاتَيْن.
انتبهوا: لا توجد عمليّة حسابيّة قبل العدد الأوّل في الحساب (عدد الرموز المطلوب هو len–1).
ملاحظة: إذا كان أقلّ من len أعداد من حلقة معيّنة حتّى نهاية السلسلة – يتمّ الحساب حتّى نهاية السلسلة.

مثال: بالنسبة للسلسلتَيْن المعروضتَيْن في المثال أعلاه، وَ **len = 3** وَ **val = 0**، تُعيد العمليّة **true**، لأنّه بدءًا **من الحلقة الثانية** في سلسلة الأعداد **ومن الحلقة الثانية** في سلسلة الرموز، توجد 3 أعداد بتسلسل نتيجةُ حسابِها هي 0 (5 – 2 – 3 = 0).



مثال إضافيّ: بالنسبة للسلسلتَيْن المعروضتَيْن في المثال أعلاه، وَ **len = 3** وَ **val = 10**، تُعيد العمليّة **true**، لأنّه بدءًا **من الحلقة الرابعة** في سلسلة الأعداد **ومن الحلقة الرابعة** في سلسلة الرموز وحتّى نهاية سلسلة الأعداد، نتيجة الحساب هي 10 (3 + 7 = 10).



/ يتبع في صفحة 6 /

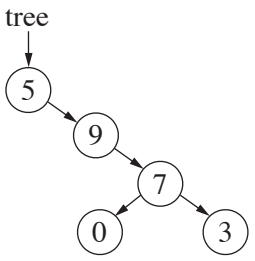
3. انتبهوا: لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 6-7 ، وبلغة **C#** في الصفحتين 8-9.

للّذين يحلُّون بلغة Java

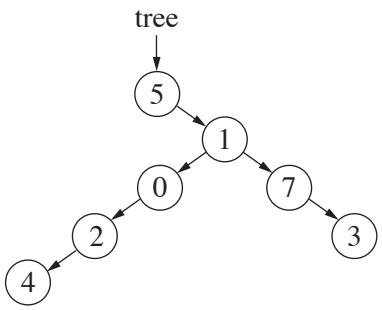
أ. معطاة العمليّة foo التي أمامكم:

```
public static int foo (BinNode<Integer> tree) {
    if (tree.getLeft() == null && tree.getRight() == null)
        return 0;
    if (tree.getLeft() == null)
        return foo (tree.getRight()) + 1;
    if (tree.getRight() == null)
        return foo (tree.getLeft()) − 1;
    return foo (tree.getLeft()) + foo (tree.getRight());
}
```

(1) نفِّذوا متابعة للعمليّة foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(2) نفِّذوا متابعة للعمليّة foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(3) بالنسبة لشجرة فيها 6 عُقَد، ما هو أكبر عدد يمكن أن تُعيده العمليّة foo ؟ اعرضوا الشجرة.

(4) بالنسبة لشجرة فيها 6 عُقَد، ما هو أصغر عدد يمكن أن تُعيده العمليّة foo ؟ اعرضوا الشجرة.

(5) اكتبوا باختصار ماذا تُعيد العمليّة foo بالنسبة لشجرة ما ليست فارغة.

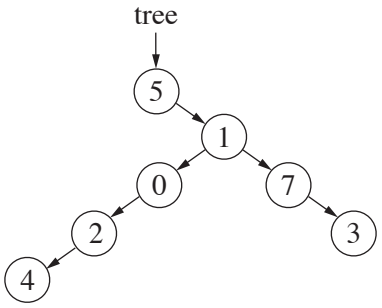
(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

ب. معطاة العمليّة goo التي أمامكم:

```
public static boolean goo (BinNode<Integer> tree) {
    if (tree == null)
        return true;
    if (foo (tree) != 0)
        return false;
    return goo (tree.getLeft()) && goo (tree.getRight());
}
```

(1) نفِّذوا متابعة للعمليّة goo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.
في هذا البند، لا حاجة لعرض متابعة للعمليّة foo .



(2) اعرضوا شجرة يوجد لها أكثر من 3 عُقَد بالنسبة لها تُعيد العمليّة goo نتيجة عكسيّة للنتيجة التي نتجت في البند الفرعيّ "ب (1)".

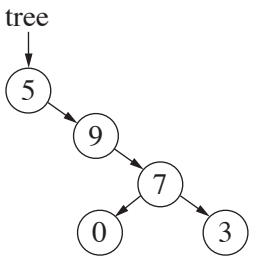
/ يتبع في صفحة 8 /

للّذين يحلُّون بلغة #C

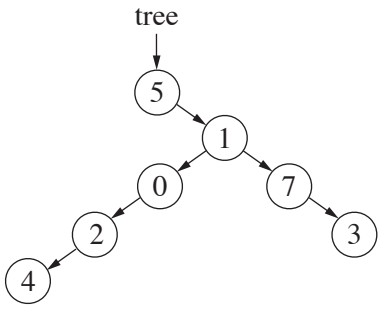
أ. معطاة العمليّة Foo التي أمامكم:

```
public static int Foo (BinNode<int> tree)
{
    if (tree.GetLeft() == null && tree.GetRight() == null)
        return 0;
    if (tree.GetLeft() == null)
        return Foo (tree.GetRight()) + 1;
    if (tree.GetRight() == null)
        return Foo (tree.GetLeft()) – 1;
    return Foo (tree.GetLeft()) + Foo (tree.GetRight());
}
```

(1) نفِّذوا متابعة للعمليّة Foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(2) نفِّذوا متابعة للعمليّة Foo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.



(3) بالنسبة لشجرة فيها 6 عُقَد، ما هو أكبر عدد يمكن أن تُعيده العمليّة Foo ؟ اعرضوا الشجرة.

(4) بالنسبة لشجرة فيها 6 عُقَد، ما هو أصغر عدد يمكن أن تُعيده العمليّة Foo ؟ اعرضوا الشجرة.

(5) اكتبوا باختصار ماذا تُعيد العمليّة Foo بالنسبة لشجرة ما ليست فارغة.

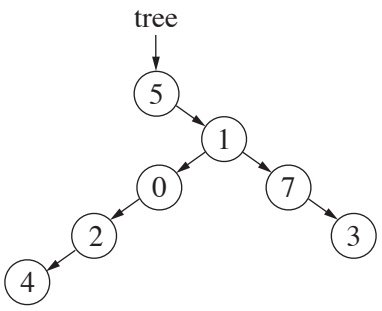
(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 9 /

ب. معطاة العمليّة Goo التي أمامكم:

```
public static bool Goo (BinNode<int> tree)
{
    if (tree == null)
        return true;
    if (Foo (tree) != 0)
        return false;
    return Goo (tree.GetLeft()) && Goo (tree.GetRight());
}
```

(1) نفِّذوا متابعة للعمليّة Goo وللشجرة التي أمامكم، واكتبوا ماذا تُعيد العمليّة. يجب عرض المتابعة.
في هذا البند، لا حاجة لعرض متابعة للعمليّة Foo .



(2) اعرضوا شجرة يوجد لها أكثر من 3 عُقَد بالنسبة لها تُعيد العمليّة Goo نتيجة عكسيّة للنتيجة التي نتجت في البند الفرعيّ "ب (1)".

/ يتبع في صفحة 10 /

الفصل الثاني (50 درجة)

في هذا الفصل أسئلة في ثلاثة مسارات:
ألغوريثمات، في الصفحات 10-14.
موديلات حسابيّة، في الصفحات 15-17.
برمجة موجّهة كائنات (بلغة Java وبلغة C#)، في الصفحات 18-27.
يجب الإجابة عن سؤالين في المسار الذي تعلّمتموه. (لكلّ سؤال – 25 درجة.)

ألغوريثمات

4. أمامكم بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. أمامكم سبعة ادّعاءات (1)–(7). اختاروا خمسة منها، واكتبوا أرقامها. اذكروا بالنسبة لكلّ ادّعاء اخترتموه هل هو صحيح أم غير صحيح. إذا كان الادّعاء صحيحًا – علِّلوا لماذا، وإذا كان غير صحيح – أعطوا مثالًا مناقضًا من رسم بيانيّ فيه أربعة رؤوس على الأقلّ.

(1) معطى رسم بيانيّ G ليس موجّهًا. إذا كان في الرسم البيانيّ قوس بين الرأسين x ، y ، فإنّ المركّبات القابلة للرَّبْط لِـ x ولِـ y هي متطابقة.

(2) معطى رسم بيانيّ G موجّه. إذا كان في الرسم البيانيّ قوس من الرأس y إلى الرأس x وقوس من الرأس x إلى الرأس y ، فإنّ المركّبات القابلة للرَّبْط جيّدًا لِـ x لِـ y هي متطابقة.

(3) في الرسم البيانيّ الموجّه الذي فيه دائرة، يوجد على الأقلّ رأسان درجة دخولهما هي أكبر من 0 .

(4) إذا كان لكلّ قوس وَزْن مختلف في رسم بيانيّ موزون قابل للرَّبْط ليس موجّهًا، عندها يكون المسار الأقصر (الأقلّ وَزْنًا) بين كلّ رأسين في الرسم البيانيّ هو مسار وحيد (أي لا يوجد أكثر من مسار واحد هو أقصر بين رأس وآخر).

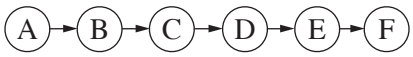
(5) في الرسم البيانيّ الموجَّه الذي فيه وَزْن كلّ قوس هو وزن مختلف، توجد شجرة امتداديّة صغرى وحيدة.

(6) لا يوجد رسم بيانيّ موجّه موزون، فيه ألغوريثم ديكسترا من الرأس y إلى الرأس x يعطي نفس المسار الأقصر (الأقلّ وَزْنًا) الذي يعطيه الألغوريثم BFS .

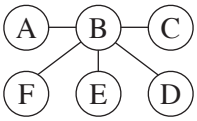
(7) يمكن عمل ترتيب طوبولوجيّ بالنسبة لكلّ رسم بيانيّ موجّه.

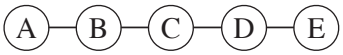
(انتبهوا: تكملة السؤال في الصفحة التالية.)

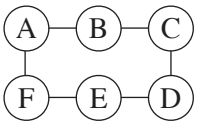
ب. شجرة امتداديّة "**سلسلة**" هي شجرة فيها لكلّ عقدة ابن واحد على الأكثر.
مثال لشجرة امتداديّة "**سلسلة**".

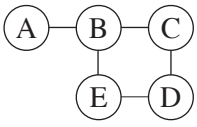


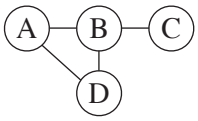
رسم بيانيّ قابل للرَّبْط ليس موجّهًا يُسمّى "**DFS سلسليّ**" إذا كانت تَنْتُج شجرة امتداديّة "**سلسلة**" في كلّ مَسْح DFS من كلّ عقدة في الرسم البيانيّ.
أمامكم سبعة رسوم بيانيّة G_1–G_7. اختاروا أربعة منها، واكتبوا بالنسبة لكلّ رسم بيانيّ اخترتموه، هل هو رسم بيانيّ "**DFS سلسليّ**" أم لا. إذا كتبتم لا – اعرضوا مَسْح DFS يَنْتُج بالنسبة له رسم بيانيّ ليس شجرة امتداديّة "**سلسلة**".

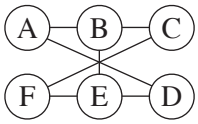
G_1:


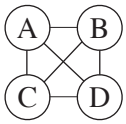
G_2:


G_3:


G_4:


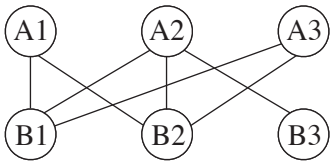
G_5:


G_6:


G_7:


/ يتبع في صفحة 12 /

5. أمامكم رسم بيانيّ ليس موجّهًا له جانبان $G = (A, B, E)$.



العُقَد في الجانب A	$\mathbf{A} = \{A1, A2, A3\}$
العُقَد في الجانب B	$\mathbf{B} = \{B1, B2, B3\}$
الأقواس بين A وَ B	$\mathbf{E} = \{(A1-B1), (A1-B2), (A2-B1), (A2-B2), (A2-B3), (A3-B1), (A3-B2)\}$

أ. العقدة M معرَّفة "جارة" للعقدة U ، إذا كان بينهما قوس.
أمامكم جدول جيرات، يعرض جارات كلّ عقدة في الرسم البيانيّ G . العُقَد الجارات لِـ A1 وَلِـ B1 معطاة في الجدول.
أكمِلوا الجدول.

U	A1	A2	A3	B1	B2	B3
M	B1 , B2			A1 , A2 , A3		

ب. معطاة قائمة لقسم من الأقواس في الرسم البيانيّ G : $L = \{(A1-B1), (A2-B2)\}$.
"**عقدة مشغولة**" هي عقدة تظهر في القائمة L (العُقَد B2، B1، A2، A1 تظهر، ولذلك هي "عُقَد مشغولة").
"**عقدة حرّة**" هي عقدة لا تظهر في القائمة L (العقدتان B3، A3 لا تظهران، ولذلك هما "عقدتان حرّتان").

معطى الألغوريثم **Path** (G, L) .

مرحلة ابتداء بيانات في الألغوريثم:

1. نُعرِّف دَوْرًا – q ، نُدخِل إليه العقدة A3 .

انتبهوا: خلال تشغيل الألغوريثم، تَدخُل إلى الدَّوْر عُقَد إضافيّة تنتمي إلى الجانب A أو إلى الجانب B .

2. نُعرِّف مصفوفة – visited بكِبَر 6 ، نُشير فيها إلى جميع العُقَد التي زُرْناها خلال تشغيل الألغوريثم (في البداية في الخليّة A3 تظهر القيمة نعم، وفي بقيّة الخلايا في المصفوفة تظهر القيمة لا).

visited:

A1	A2	A3	B1	B2	B3
لا	لا	نعم	لا	لا	لا

3. نعرِّف مصفوفة – parent بكِبَر 6 ، نَحفَظ فيها بالنسبة لكلّ عقدة "والدها"، أي العقدة التي وصلنا منها إلى هذه العقدة خلال تشغيل الألغوريثم (في البداية جميع الخلايا في المصفوفة هي فارغة).

parent:

A1	A2	A3	B1	B2	B3

(انتبهوا : تكملة السؤال في الصفحة التالية .)

مرحلة تشغيل الألغوريثم:

1. طالما كان الدَّوْر – q ليس فارغًا:
   - نُخرِج العقدة الموجودة في رأس الدَّوْر، ونُسمّيها U .
   - نفحص هل U موجودة في الجانب B وهي أيضًا "عقدة حرّة":
     - **إذا كانت كذلك** – يُعيد الألغوريثم قائمة عُقَد هي: U ووالد U ووالد والد U وهكذا فصاعدًا، حتّى العقدة التي ليس لها والد (أي العقدة التي خليّتُها فارغة في المصفوفة parent)، **وينتهي الألغوريثم**.
     - **إذا لم تكن كذلك** (أي أنّه إذا كانت U موجودة في الجانب A أو إذا كانت U هي "عقدة مشغولة") – بالنسبة لكلّ عقدة جارة – M للعقدة U :

     إذا **لم نكن قد زُرْنا بَعْدُ** العقدة M (لا = visited[M])، وكذلك يتحقّق **أحد الشرطين** التاليين:
     – U في الجانب **A** والقوس (U–M) **ليس موجودًا** في القائمة L
     – U في الجانب **B** والقوس (U–M) **موجود** في القائمة L

     عندها:
     أ. نُدخِل M إلى الدَّوْر.
     ب. نكتب القيمة **نعم** في الخليّة M في المصفوفة visited .
     جـ. نُحدِّث أنّ العقدة U هي والد M في المصفوفة parent .
2. إذا كان الدَّوْر فارغًا، نُعيد null ، **وينتهي الألغوريثم**.

تتبّعوا بواسطة جدول متابعة تشغيل الألغوريثم Path على الرسم البيانيّ المعطى G ، مع القائمة L .
يجب أن تشمل المتابعة في كلّ تكرار البنود التالية:
- حالة الدَّوْر
- العقدة U التي أخرجناها من الدَّوْر
- المصفوفة visited
- المصفوفة parent

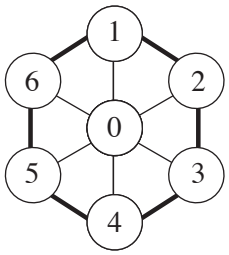
الدَّوْر	U	المصفوفة visited						المصفوفة parent					
		A1	A2	A3	B1	B2	B3	A1	A2	A3	B1	B2	B3
←[ ]													
←[ ]													

جـ. اكتبوا **قائمة العُقَد** التي يُعيدها الألغوريثم.

/ يتبع في صفحة 14 /

6. الرسم البيانيّ "**كوكب مغلق**" – G_n هو رسم بيانيّ ليس موجّهًا مكوَّنٌ من دائرة فيها n رؤوس (من 1 حتّى n)، تُسمّى هذه الرؤوس "**رؤوس الدائرة**"، ومن رأس إضافيّ – 0 يُسمّى "**رأس مركز**"، موجود في مركز الدائرة ومربوط بكلّ واحد من "رؤوس الدائرة" (بحيث يوجد بالمجمَل $\mathbf{n+1}$ رؤوس في الرسم البيانيّ). عدد "رؤوس الدائرة" في الرسم البيانيّ هو أكبر من 2 (أي $n > 2$).

مثال: الرسم البيانيّ الذي أمامكم هو رسم بيانيّ "**كوكب مغلق**" – G_6، لأنّ الرؤوس 1 حتّى 6 تُكوِّن دائرة، والرأس الإضافيّ، 0، موجود في المركز ومربوط بكلّ واحد من هذه الرؤوس.



"**أقواس الدائرة**" هي الأقواس التي تربط "رؤوس الدائرة" الواحد بالآخر (مثلًا في المثال أعلاه، القوس الذي يربط الرأسين 1 وَ 2 فيما بينهما، والقوس الذي يربط الرأسين 2 وَ 3 فيما بينهما وهكذا).

"**أقواس مركزيّة**" هي الأقواس التي تربط "رأس مركز" بكلّ واحد من "رؤوس الدائرة" (مثلًا في المثال أعلاه، القوس الذي يربط الرأس 0 بالرأس 1، والقوس الذي يربط الرأس 0 بالرأس 2 وهكذا).

أ. معطى رسم بيانيّ "**كوكب مغلق**" – G_n.

(1) ما هي درجات "رؤوس الدائرة" في الرسم البيانيّ، وما هي درجة "رأس مركز"؟ علِّلوا إجابتكم.

(2) كم "قوس دائرة" يوجد في الرسم البيانيّ، وكم "قوسًا مركزيًّا" يوجد في الرسم البيانيّ؟ علِّلوا إجابتكم.

(3) هل الرسم البيانيّ له جانبان؟ علِّلوا إجابتكم.

ب. معطى رسم بيانيّ موزون هو "**كوكب مغلق**" – G_n، فيه لجميع "أقواس الدائرة" وَزْن 2، ولجميع "الأقواس المركزيّة" وَزْن 1. كم شجرة امتداديّة صغرى مختلفة توجد في الرسم البيانيّ؟ علِّلوا إجابتكم.

جـ. معطى رسم بيانيّ موزون هو "**كوكب مغلق**" – G_n، فيه لجميع "أقواس الدائرة" وَزْن b (أكبر من 0)، ولجميع "الأقواس المركزيّة" وَزْن m (أكبر من 0).

(1) عوِّضوا قيمتين لِـ b وَ m، بالنسبة لهما كلّ مسار أقصر (أقلّ وَزْنًا) بين "رأسَي دائرة" يمرّ عَبْرَ "رأس مركز". علِّلوا إجابتكم.

(2) معطى أنّ $b = 10$. ما هي أكبر قيمة لِـ m التي بالنسبة لها كلّ مسار أقصر (أقلّ وَزْنًا) بين "رأسَي دائرة" لا يزال يمرّ عَبْرَ "رأس مركز"؟ علِّلوا إجابتكم.

/ يتبع في صفحة 15 /

موديلات حسابيّة

7. أمامكم بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. **(1)** أمامكم ثلاث لغات $L_3 - L_1$ فوق الأبجديّة $\{a, b, c\}$:

$$L_1 = \{a^n b^k c^n \mid 0 < n < 1000 < k\}$$

$$L_2 = \{a^n b^k c^n \mid 0 < k < 1000 < n\}$$

$$L_3 = \{a^n b^k c^n \mid 0 < k < 1000 < n < 2000\}$$

بالنسبة لكلّ واحدة من اللُّغات $L_3 - L_1$ ، اكتبوا هل هي نظاميّة أم لا. علِّلوا إجابتكم (يكفي شرح كلاميّ؛ لا حاجة إلى أوتومات).

(2) أمامكم ثلاث لغات $L_6 - L_4$ فوق الأبجديّة $\{a, b, c\}$:

$$L_4 = L_2 \cap R(L_2)$$

$$L_5 = L_1 \cup L_2$$

$$L_6 = \overline{L_1} \cup \overline{L_3}$$

بالنسبة لكلّ واحدة من اللُّغات $L_6 - L_4$ ، اكتبوا هل هي نظاميّة أم لا. علِّلوا إجابتكم (يكفي شرح كلاميّ؛ لا حاجة إلى أوتومات).

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

$$L = \{a^n b^m a^k \mathbf{w} \mid n, m, k > 0\}$$

w هي كلمة فوق الأبجديّة $\{b, c\}$ معرَّفة بالطريقة التالية:

- إذا كان $n + k$ يقسم على 3 بدون باقٍ، فإنّ w هي كلمة فارغة .
- إذا كان $n + k$ يقسم على 3 مع باقي 1 ، فإنّ w هي كلمة لا تحوي الحرف b بتاتًا.
- إذا كان $n + k$ يقسم على 3 مع باقي 2 ، فإنّ w هي كلمة لا تحوي الحرف c بتاتًا.

مثال لكلمة تنتمي إلى اللُّغة L : $\overbrace{a}^{a^n}\overbrace{bb}^{b^m}\overbrace{aaa}^{a^k}\overbrace{ccc}^{w}$

الشرح: $n = 1$ ، $k = 3$. $n + k$ يقسم على 3 مع باقي 1 ، وَ w هي كلمة لا تحوي الحرف b .

ابنوا أوتوماتًا نهائيًّا محدودًا ليس كاملًا يتلقّى اللُّغة L .

/ يتبع في صفحة 16 /

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

$$L = \{a^n b^m c^k \mid \begin{cases} n\%2 = 0: & m\%2 = 0,\ n = m/2 + k,\ n,m,k > 0 \\ n\%2 = 1: & n \le m + k - 2 \end{cases} \}$$

مثال لكلمة في اللُّغة L بالنسبة لِـ n **زوجي**: $\overbrace{a\,a\,a\,a}^{n=4}\ \overbrace{b\,b}^{m=2}\ \overbrace{c\,c\,c}^{k=3}$.

الشرح: $n\%2 = 0$ ، ووفقًا لذلك $m\%2 = 0$ أيضًا، ويتحقّق $4 = (2 / 2) + 3$ وأيضًا $n, m, k > 0$.

مثال لكلمة في اللُّغة L بالنسبة لِـ n **فرديّ**: $\overbrace{a\,a\,a}^{n=3}\ \overbrace{b\,b\,b}^{m=3}\ \overbrace{c\,c\,c\,c}^{k=4}$.

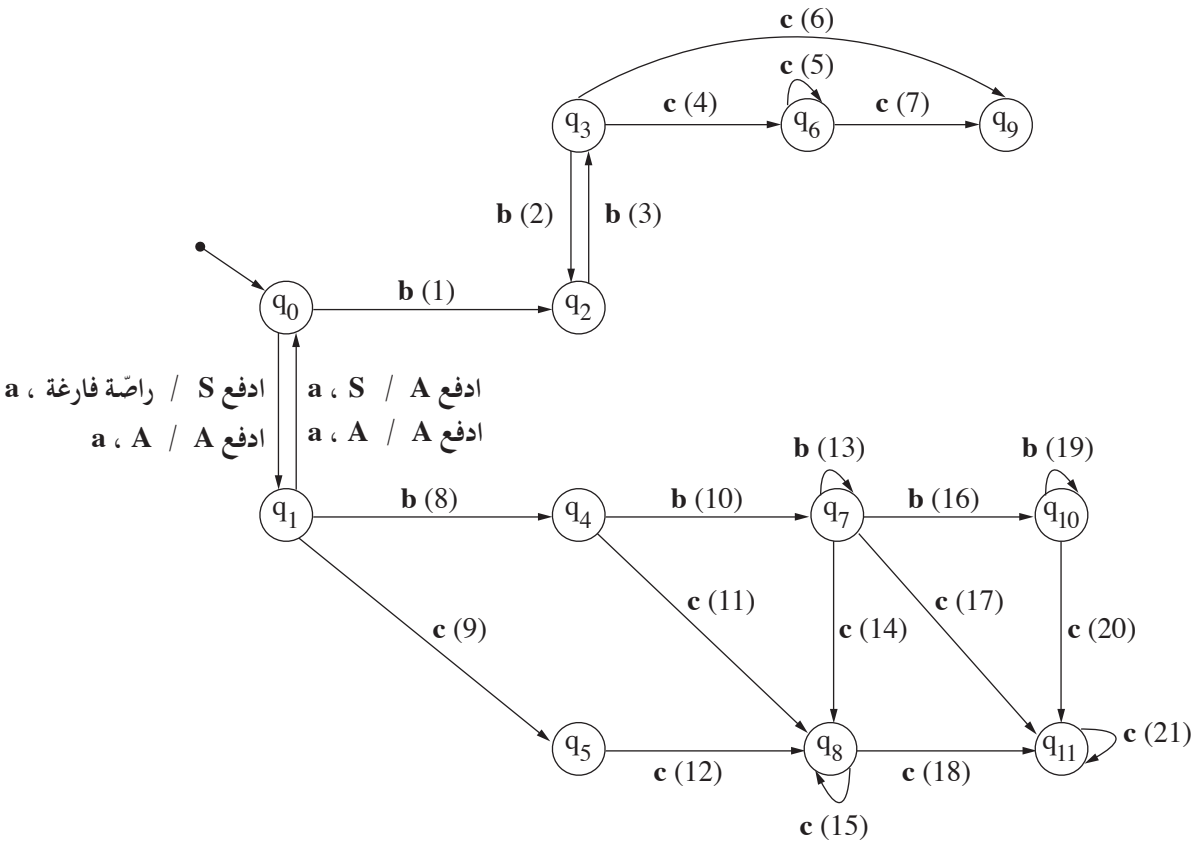
الشرح: $n\%2 = 1$ ووفقًا لذلك يتحقّق $3 \le 3 + 4 - 2$.

أ. أمامكم خمس كلمات. بالنسبة لكلّ واحدة منها، اذكروا هل الكلمة تنتمي إلى اللُّغة L . علِّلوا إجابتكم.

abcc, acccc, aabbc, aabbbb, aaabbc .

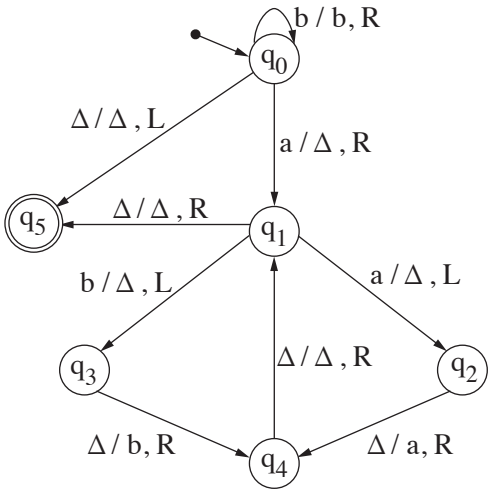
ب. اكتبوا أقصر كلمة في اللُّغة L بالنسبة لِـ n زوجيّ، وإحدى أقصر الكلمات بالنسبة لِـ n فرديّ (توجد أكثر من إمكانيّة واحدة) .

جـ. أمامكم أوتومات **راصّة محدود** يتلقّى اللُّغة L . توجد في الأوتومات جميع الحالات والانتقالات وإشارات المُدخَلات المطلوبة. لكن باستثناء الانتقالات بين q_0 وَ q_1 ، ينقص في الانتقالات الحرف الذي في رأس الراصّة والعمليّة على الراصّة (هذه الانتقالات مُرقَّمة من 1 حتّى 21)، إضافةً إلى ذلك، لم تتمّ الإشارة إلى الحالات المتلقّية في الأوتومات.
بالنسبة لكلّ انتقال مُرقَّم، اذكروا رقمه وأكمِلوا الإشارة الناقصة: الحرف الذي في رأس الراصّة والعمليّة التي على الراصّة (ادفع/push أو اسحب/pop أو بدون تغيير). إضافةً إلى ذلك، اكتبوا ما هي الحالات المتلقّية في الأوتومات (لا حاجة لنسخ الأوتومات إلى دفتركم).



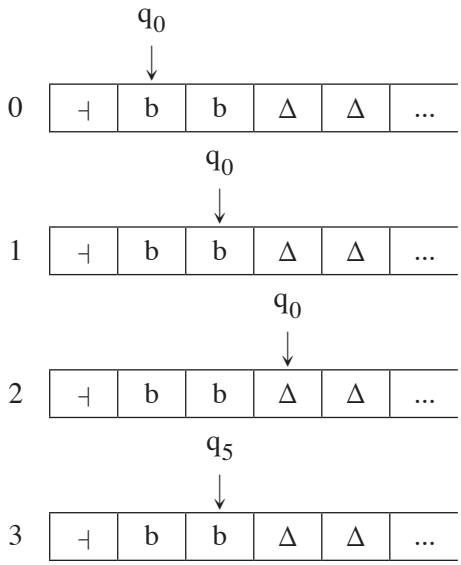
/ يتبع في صفحة 17 /

9. أمامكم آلة تيورينج تتلقّى كلمات فوق الأبجديّة {a, b} (مفتاح: R – يمين، L – يسار).



في البند "أ" الذي أمامكم يُطلَب منكم تنفيذ متابعة لعمل الآلة. في المتابعة، يجب أن تُبيِّنوا الشريط في كلّ مرحلة، وفوق أيّة خليّة يقع الرأس القارئ، وفي أيّة حالة q_5–q_0 موجودة.

فيما يلي مثال للطريقة التي يجب فيها تنفيذ متابعة بالنسبة للآلة المعطاة والكلمة bb.



أ. **(1)** تتبّعوا عمل الآلة بالنسبة للكلمة a . يجب عرض المتابعة.

(2) تتبّعوا عمل الآلة بالنسبة للكلمة aba . يجب عرض المتابعة.

(3) اعرضوا الشريط الذي يَنتُج في نهاية العمل بالنسبة للكلمة baba . لا حاجة لعرض المتابعة.

ب. **(1)** هل توجد كلمة تحوي a وَ b أيضًا، لا تتغيّر بعد تشغيل الآلة عليها؟
إذا كانت إجابتكم نعم – اكتبوا مثالًا لمثل هذه الكلمة ، إذا كانت إجابتكم لا – علِّلوا إجابتكم.

(2) اكتبوا باختصار ماذا تعمل الآلة.

/ يتبع في صفحة 18 /

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 الصفحة 18، وبلغة **#C** في الصفحة 19.

للّذين يحلُّون بلغة Java

في أكاديميّة الموسيقى "**أغنية جديدة**"، طُوِّرت منظومة مُحَوْسَبة فيها الفئات التالية:
Instrum – آلة عَزْف، **Bow** – آلة وتريّة ذات قوس، **Wind** – آلة نَفْخ، **Perc** – آلة إيقاعيّة، **Musician** – عازف في الفرقة، **Orchestra** – فرقة.

فيما يلي تفصيل لصفات الفئات:

للفئة **Instrum** (آلة عَزْف) صفتان: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ).

للفئة **Bow** (آلة وتريّة ذات قوس) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، bowLength – طول القوس (من نمط صحيح).

للفئة **Wind** (آلة نَفْخ) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، numberOfValves – عدد الصمامات في الآلة (من نمط صحيح).

للفئة **Perc** (آلة إيقاعيّة) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، hasSticks – هل يوجد استعمال لعصيّ النَّقْر (من نمط بولياني. إذا كان يوجد استعمال للعصيّ – true ، وإذا لم يوجد – false).

للفئة **Musician** (عازف في الفرقة) صفتان: name – اسم العازف (من نمط نصّ)، instrum – آلة عزفه (من نمط **Instrum**). كلّ عازف يعزف على آلة واحدة فقط.

للفئة **Orchestra** (فرقة) صفتان: arrM – مصفوفة للعازفين (من نمط **Musician**)، count – عدد العازفين المحفوظين في المصفوفة (من نمط صحيح).

يظهر العازفون بتسلسُل من بداية المصفوفة بدون null بينهم، لكن يمكن أن لا تكون المصفوفة مليئة حتّى النهاية.

أ. **(1)** ارسموا مخطّطًا هرميًّا بين جميع الفئات.
يجب الإشارة إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ——◆.
(2) اكتبوا عناوين الفئات وصفاتها. افترِضوا أنّه عُرِّفت عمليّات get وَ set وعمليّات بنائيّة في كلّ فئة.

ب. اكتبوا في الفئة **Orchestra** عمليّة اسمها withSticks، تُعيد عدد العازفين في الفرقة الذين يعزفون على آلات إيقاعيّة تُستعمَل فيها عصيّ.

جـ. مصفوفة arrM "مرتّبة" هي مصفوفة يظهر فيها أوّلًا العازفون على الآلات الوتريّة ذات القوس، وبعد ذلك العازفون على آلات النَّفْخ، وبعد ذلك العازفون على الآلات الإيقاعيّة.
اكتبوا في الفئة **Orchestra** عمليّة اسمها insert، تتلقّى عازفًا – musician لا يظهر في المصفوفة arrM.
افترِضوا أنّه قبل بدء العمليّة كانت المصفوفة "مرتّبة"، وأنّه يوجد في الفرقة على الأقلّ عازف واحد على كلّ نوع من الآلات.
إذا لم يكن هناك مكان لعازف إضافيّ في المصفوفة، تُعيد العمليّة false. خلاف ذلك، تُدخِل العمليّة العازف إلى المصفوفة في المكان الملائم بحيث تبقى المصفوفة "مرتّبة" (ويظهر العازفون بتسلسُل في بداية المصفوفة)، وتُعيد true.
ملاحظة: يمكن تغيير ترتيب العازفين، بشرط أن تكون المصفوفة "مرتّبة" في نهاية العمليّة.

/ يتبع في صفحة 19 /

للّذين يحلُّون بلغة #C

في أكاديميّة الموسيقى "**أغنية جديدة**"، طُوِّرت منظومة مُحَوْسَبة فيها الفئات التالية:

Instrum – آلة عَزْف، **Bow** – آلة وتريّة ذات قوس، **Wind** – آلة نَفْخ، **Perc** – آلة إيقاعيّة، **Musician** – عازف في الفرقة، **Orchestra** – فرقة.

فيما يلي تفصيل لصفات الفئات:

للفئة **Instrum** (آلة عَزْف) صفتان: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ).

للفئة **Bow** (آلة وتريّة ذات قوس) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، bowLength – طول القوس (من نمط صحيح).

للفئة **Wind** (آلة نَفْخ) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، numberOfValves – عدد الصمامات في الآلة (من نمط صحيح).

للفئة **Perc** (آلة إيقاعيّة) ثلاث صفات: name – اسم الآلة (من نمط نصّ)، material – المادّة التي صُنِعت منها الآلة (من نمط نصّ)، hasSticks – هل يوجد استعمال لعصيّ النَّقْر (من نمط بولينيّ. إذا كان يوجد استعمال للعصيّ – true، وإذا لم يوجد – false).

للفئة **Musician** (عازف في الفرقة) صفتان: name – اسم العازف (من نمط نصّ)، instrum – آلة عزفه (من نمط **Instrum**). كلّ عازف يعزف على آلة واحدة فقط.

للفئة **Orchestra** (فرقة) صفتان: arrM – مصفوفة للعازفين (من نمط **Musician**)، count – عدد العازفين المحفوظين في المصفوفة (من نمط صحيح).

يظهر العازفون بتسلسُل من بداية المصفوفة بدون null بينهم، لكن يمكن أن لا تكون المصفوفة مليئة حتّى النهاية.

أ. **(1)** ارسموا مخطّطًا هرميًّا بين جميع الفئات.

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

(2) اكتبوا عناوين الفئات وصفاتها. افترِضوا أنّه عُرِّفت عمليّات Get وَ Set وعمليّات بنائيّة في كلّ فئة.

ب. اكتبوا في الفئة **Orchestra** عمليّة اسمها WithSticks، تُعيد عدد العازفين في الفرقة الذين يعزفون على آلات إيقاعيّة تُستعمَل فيها عصيّ.

جـ. مصفوفة arrM "مرتّبة" هي مصفوفة يظهر فيها أوّلًا العازفون على الآلات الوتريّة ذات القوس، وبعد ذلك العازفون على آلات النَّفْخ، وبعد ذلك العازفون على الآلات الإيقاعيّة.

اكتبوا في الفئة **Orchestra** عمليّة اسمها Insert، تتلقّى عازفًا – musician لا يظهر في المصفوفة arrM.

افترِضوا أنّه قبل بدء العمليّة كانت المصفوفة "مرتّبة"، وأنّه يوجد في الفرقة على الأقلّ عازف واحد على كلّ نوع من الآلات.

إذا لم يكن هناك مكان لعازف إضافيّ في المصفوفة، تُعيد العمليّة false. خلاف ذلك، تُدخِل العمليّة العازف إلى المصفوفة في المكان الملائم بحيث تبقى المصفوفة "مرتّبة" (ويظهر العازفون بتسلسُل في بداية المصفوفة)، وتُعيد true.

ملاحظة: يمكن تغيير ترتيب العازفين، بشرط أن تكون المصفوفة "مرتّبة" في نهاية العمليّة.

/ يتبع في صفحة 20 /

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

<u>للّذين يحلُّون بلغة Java</u>

أمامكم ثلاثة بنود "أ – جـ"، لا علاقة بينها. <u>أجيبوا عن البنود الثلاثة</u>.

أ. أمامكم الفئة **Father** بدون تطبيق العمليّات البنائيّة، والفئة **Son** بدون تطبيق العمليّات البنائيّة، وعمليّة toString .

```
public class Father {
    private static int st = 0;
    protected int num;
    public Father () {…}
    public Father (int num) {…}
    public String toString() {return "st=" + st + ", num=" + num;}
}
-----------------------------------------------------------------
public class Son extends Father {
    private char ch;
    public Son (int num, char ch){…}
    public Son (int num) {…}
    public String toString() {... }
}
```

أمامكم قطعة كود والمُخرَج الذي نَتَجَ في أعقاب تشغيل قطعة الكود:

الكود		المُخرَج	
Father x1 = new Father ();			
System.out.println (x1);	//1	st=1, num=9	//1
Father x2 = new Father (3);			
System.out.println (x2);	//2	st=1, num=3	//2
Father x3 = new Son (5, 'A');			
System.out.println (x3);	//3	st=2, num=5, ch=A	//3
Father x4 = new Son (7);			
System.out.println (x4);	//4	st=2, num=7, ch=Z	//4

طبِّقوا العمليّات البنائيّة في الفئة Father وفي الفئة Son ، والعمليّة toString في الفئة Son ،
وفقًا لمبادئ البرمجة الموجّهة كائنات، بحيث ينتُج المخرَج المعطى.
لا تضيفوا عمليّات إضافيّة إلى الفئتين. الإجابة التي تتضمّن عمليّات إضافيّة في الفئتين، لن تحصل على درجات.

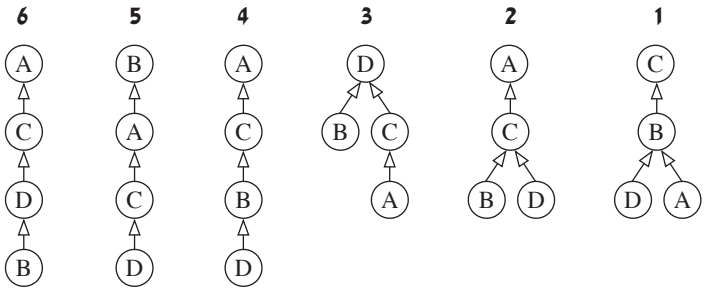
(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 21 /

ب. معطاة الفئات **A**، **B**، **C**، **D** ، والعمليّة main في الفئة **Tester** ، التي تعمل بشكل سليم.

```
public class Tester {
    public static void main (string[] args) {
        C c1 = new B();
        A a1 = new C();
        A a2 = new B();
        C c2 = new D();
    }
}
```

(1) أمامكم ستّة مخطّطات هرميّة 1–6 (السهم يشير إلى علاقة توريث). بالنسبة لكلّ واحد منها، اذكروا هل يمكن أم لا يمكن أنّه يصف العلاقة الهرميّة بين الفئات **A**، **B**، **C**، **D**. إذا ذكرتم أنّه لا يمكن، علِّلوا لماذا.



(2) في نهاية العمليّة main أُضيفَ سطرا الكود التاليان اللذان يعملان بشكل سليم:

```
A a3 = new B();
D d1 = (D)a3;
```

في أعقاب ذلك، هل تتغيّر إجابتكم في البند الفرعيّ "ب (1)"؟ علِّلوا إجابتكم.

جـ. معطاة الفئة **BBB** ، التي تَرِث من الفئة **AAA** :

```
public class BBB extends AAA {
    private char ch;
    public BBB (int num, char ch) {
        super (num);
        this.ch = ch;
    }
    public BBB (char ch) {this.ch = ch; }
    public int gg (int val) {return super.gg(val) + num; }
    public int mm () {return AAA.c + 2; }
    public boolean nn () {return this.mm (10); }
}
```

اكتبوا في الفئة AAA **صفات** الفئة (بما فيها أُذونات الوصول (הרשאות הגישה)) **وعناوين** العمليّات المطلوبة لتكون الفئة BBB سليمة، وفقًا للبرمجة الموجّهة كائنات.

لا حاجة لتطبيق العمليّات التي أضفتم عناوينها في الفئة AAA . لا تُغيِّروا الكود في الفئة BBB.

/ يتبع في صفحة 22 /

للّذين يحلُّون بلغة #C

أمامكم ثلاثة بنود "أ – جـ"، لا علاقة بينها. أجيبوا عن البنود الثلاثة.

أ. أمامكم الفئة **Father** بدون تطبيق العمليّات البنائيّة، والفئة **Son** بدون تطبيق العمليّات البنائيّة، وعمليّة ToString .

```
public class Father {
    private static int st = 0;
    protected int num;
    public Father() {…}
    public Father (int num) {…}
    public override string ToString() { return "st=" + st + ", num=" + num; }
}
public class Son : Father {
    private char ch;
    public Son (int num, char ch) ...{…}
    public Son (int num) ...{…}
    public override string ToString() {...}
}
```

أمامكم قطعة كود والمُخرَج الذي نَتَجَ في أعقاب تشغيل قطعة الكود:

الكود	المُخرَج
Father x1 = new Father();	
Console.WriteLine (x1); //1	st=1, num=9 //1
Father x2 = new Father (3);	
Console.WriteLine (x2); //2	st=1, num=3 //2
Father x3 = new Son (5, 'A');	
Console.WriteLine (x3); //3	st=2, num=5, ch=A //3
Father x4 = new Son (7);	
Console.WriteLine (x4); //4	st=2, num=7, ch=Z //4

طبِّقوا العمليّات البنائيّة في الفئة Father وفي الفئة Son ، والعمليّة ToString في الفئة Son ،
وفقًا لمبادئ البرمجة الموجّهة كائنات، بحيث ينتُج المخرَج المعطى.
لا تضيفوا عمليّات إضافيّة إلى الفئتين. الإجابة التي تتضمّن عمليّات إضافيّة في الفئتين، لن تحصل على درجات.

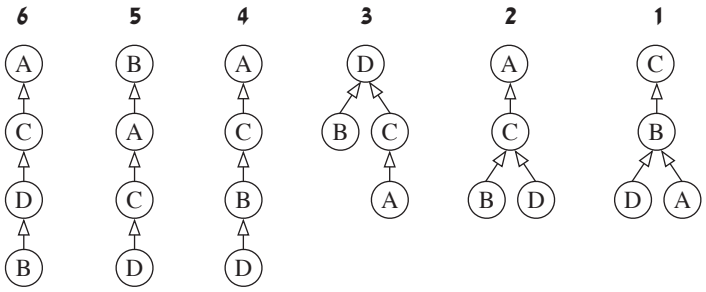
(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 23 /

ب. معطاة الفئات **A**، **B**، **C**، **D** ، والعمليّة Main في الفئة **Tester** ، التي تعمل بشكل سليم.

```
public class Tester {
    public static void Main (string[] args) {
        C c1 = new B();
        A a1 = new C();
        A a2 = new B();
        C c2 = new D();
    }
}
```

(1) أمامكم ستّة مخطّطات هرميّة 1–6 (السهم يشير إلى علاقة توريث). بالنسبة لكلّ واحد منها، اذكروا هل يمكن أم لا يمكن أنّه يصف العلاقة الهرميّة بين الفئات **A**، **B**، **C**، **D**. إذا ذكرتم أنّه لا يمكن، علِّلوا لماذا.



(2) في نهاية العمليّة Main أُضيفَ سطرا الكود التاليان اللذان يعملان بشكل سليم:

```
A a3 = new B();
D d1 = (D)a3;
```

في أعقاب ذلك، هل تتغيّر إجابتكم في البند الفرعيّ "ب (1)"؟ علِّلوا إجابتكم.

جـ. معطاة الفئة **BBB** ، التي تَرِث من الفئة **AAA** :

```
public class BBB : AAA {
    private char ch;
    public BBB (int num, char ch) : base (num) {
        this.ch = ch;
    }
    public BBB (char ch) {this.ch = ch;}
    public override int GG (int val) {return base.GG (val) + num;}
    public int MM() {return AAA.c + 2;}
    public bool NN() {return this.MM (10);}
}
```

اكتبوا في الفئة AAA **صفات** الفئة (بما فيها أُذونات الوصول (הרשאות הגישה)) **وعناوين** العمليّات المطلوبة لتكون الفئة BBB سليمة، وفقًا للبرمجة الموجّهة كائنات.

لا حاجة لتطبيق العمليّات التي أضفتم عناوينها في الفئة AAA . لا تُغيِّروا الكود في الفئة BBB.

/ يتبع في صفحة 24 /

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

للّذين يحلُّون بلغة Java

أمامكم الفئات **A**، **B** ، **C** :

```
public class A {
    public int num;

    public A() {
        this.num = 1;
    }
    public A(int num) {
        this.num = num;
    }
    public int func (int val) {
        return val * 2;
    }
    public int func (B other) {
        return this.num – other.num;
    }
}
```

```
public class B extends A {
    public B() {
        super(2);
    }
    public B(int val) {
        super(val);
        this.num = this.num + func (val);
    }
    public int func (A other) {
        return this.num + (other.num * 10);
    }
    public int func (B other) {
        return this.num + other.num;
    }
}
```

```
public class C extends B {
    public C(int val)  {
        this.num = this.num – val;
    }
    public int func (B other)  {
        return this.num * other.num;
    }
    public int func (C c)  {
        return this.num * 4;
    }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

أ. أمامكم قطعة كود تعمل بدون أخطاء:

```
A a = new A();
A ab = new B();
A ac = new C(0);
B b = new B(1);
B bc = new C(1);
C c = new C(1);
```

اعرضوا الكائنات التي تكوَّنت (نوع الكائن، ونوع التوجيه، وقيمة الصفة num).

ب. أمامكم تكملة قطعة الكود (مرقَّمة) التي تعمل بدون أخطاء.
اذكروا رقم سطر الطباعة، واكتبوا ماذا يَطْبَع الكود.

```
1.  System.out.println (a.func (b));
2.  System.out.println (a.func (c));
3.  System.out.println (ab.func ((B)ab));
4.  System.out.println (((B)ab).func (ab));
5.  System.out.println (ab.func (10));
6.  System.out.println (a.func (a.func (2)));
7.  System.out.println (ac.func (c));
8.  System.out.println (c.func (ac));
9.  System.out.println (c.func (a));
10. System.out.println (((C)bc).func ((C)bc));
```

/ يتبع في صفحة 26 /

للّذين يحلُّون بلغة #C

أمامكم الفئات **A**، **B** ، **C** :

```
public class A {
    public int num;

    public A()
    {
        this.num = 1;
    }
    public A (int num)
    {
        this.num = num;
    }
    public int Func (int val)
    {
        return val * 2;
    }
    public virtual int Func (B other)
    {
        return this.num - other.num;
    }
}
```

```
public class B : A {
    public B() : base (2) { }
    public B(int val) : base (val)
    {
        this.num = this.num + Func (val);
    }
    public int Func (A other)
    {
        return this.num + (other.num * 10);
    }
    public override int Func (B other)
    {
        return this.num + other.num;
    }
}
```

```
public class C : B {
    public C (int val)
    {
        this.num = this.num - val;
    }
    public override int Func (B other)
    {
        return this.num * other.num;
    }
    public int Func (C c)
    {
        return this.num * 4;
    }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 27 /

أ. أمامكم قطعة كود تعمل بدون أخطاء:

```
A a = new A();
A ab = new B();
A ac = new C(0);
B b = new B(1);
B bc = new C(1);
C c = new C(1);
```

اعرضوا الكائنات التي تكوَّنت (نوع الكائن، ونوع التوجيه، وقيمة الصفة num).

ب. أمامكم تكملة قطعة الكود (مرقَّمة) التي تعمل بدون أخطاء.
اذكروا رقم سطر الطباعة، واكتبوا ماذا يَطْبَع الكود.

```
1.  Console.WriteLine (a.Func (b));
2.  Console.WriteLine (a.Func (c));
3.  Console.WriteLine (ab.Func ((B)ab));
4.  Console.WriteLine (((B)ab).Func (ab));
5.  Console.WriteLine (ab.Func (10));
6.  Console.WriteLine (a.Func (a.Func (2)));
7.  Console.WriteLine (ac.Func (c));
8.  Console.WriteLine (c.Func (ac));
9.  Console.WriteLine (c.Func (a));
10. Console.WriteLine (((C)bc).Func ((C)bc));
```

בהצלחה!
نتمنّى لكم النّجاح!