מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: **מועד חלופי**, קיץ תשפ"ו, 2026
מספר השאלון: 899271
תרגום לערבית (2)

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: **موعد بديل**، صيف 2026
رقم النّموذج: 899271
ترجمة إلى العربيّة (2)

מדעי המחשב

הוראות

א. משך הבחינה: שלוש שעות וחצי.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שני פרקים.
יש לבחור בארבע שאלות סך הכול.
לכל שאלה — 27 נקודות; תלמיד שצבר מעל 100 נקודות יקבל 100.
סך הכול — 100 נקודות.
בפרק השני יש לענות על שאלות במסלול אחד בלבד.

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

ד. הוראות מיוחדות:
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 ونصف.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج فصلان.
يجب اختيار ما مجموعه أربعة أسئلة.
لكلّ سؤال – 27 درجة؛ الطّالب الّذي جَمَّع أكثر من 100 درجة يحصل على 100.
المجموع – 100 درجة.
يجب الإجابة في الفصل الثّاني عن أسئلة من مسار واحد فقط.

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

د. تعليمات خاصّة:
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 أربعة أسئلة من ستّة الأسئلة في النّموذج.

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 بينارية وحلقة، بدون تطبيقها. إذا استعملتم عمليّات إضافيّة، يجب تطبيقها.

الفصل الأوّل

1. معطاة الفئة **DigitCount**، ولها صفتان:

الشرح	الصفة
قيمة الرقم (0-9)	private int digit;
عدد مرّات ظهور الرقم	private int count;

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

كما توجد في الفئة العمليّة التالية:

الشرح	العمليّة

عمليّة بنائيّة	public DigitCount (int digit, int count) {
	this.digit = digit;
	this.count = count;
	}

أ. طبِّقوا العمليّة التي أمامكم:

Java – public static Node<DigitCount> buildList (int num)

C# – public static Node<DigitCount> BuildList (int num)

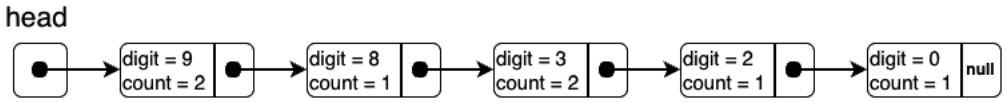
تتلقّى العمليّة عددًا صحيحًا موجبًا num أكبر من 0 ، وتُعيد قائمة مرتبطة من نوع DigitCount.

بالنسبة لكلّ رقم (0-9) في العدد num ، يُحفَظ في القائمة كائن من الفئة DigitCount الذي قِيَمُهُ الرقم نفسه وعدد مرّات ظهوره في العدد num. تُركَّب القائمة حسب القواعد التالية:

- تكون القائمة مرتّبة حسب قيمة الأرقام بترتيب تنازليّ (من الرقم ذي القيمة الأعلى إلى الرقم ذي القيمة الأقلّ).
- كلّ رقم يظهر مرّة واحدة فقط في القائمة.
- فقط الأرقام التي تظهر في العدد num تظهر في القائمة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

مثال: بالنسبة لاستدعاء العمليّة مع num = 3389920 ، وأيضًا بالنسبة لِـ num = 9389302 ، تُعاد القائمة:



الشرح: توجد في القائمة حلقة بالنسبة لكلّ رقم يظهر في العدد num. الحلقة تُمثِّل الرقم وعدد مرّات ظهور الرقم في num.
القائمة تحوي فقط الأرقام الموجودة في num ، وهي مرتّبة حسب قيمة الأرقام من أكبر قيمة إلى أصغر قيمة.

ب. طبِّقوا العمليّة التي أمامكم:

Java: public static int buildMaxNum (int num)

C#: public static int BuildMaxNum (int num)

تتلقّى العمليّة عددًا صحيحًا num أكبر من 0 ، وتُعيد أكبر عدد يمكن تكوينه بواسطة الأرقام في العدد num ، حسب عدد مرّات ظهور كلّ رقم.
ملاحظة: يمكن استعمال العمليّة التي كتبتموها في البند "أ".

مثال: بالنسبة لِـ num = 3389920 وأيضًا بالنسبة لِـ num = 9389302 ، تُعيد العمليّة العدد: 9983320.
الشرح: هذا العدد هو العدد الأكبر الذي يمكن تكوينه بواسطة تغيير ترتيب الأرقام في العدد num.

/ يتبع في صفحة 4 /

2. معطاة الفئة **Cube** – مكعّب، ولها صفتان:

- size – كِبَر المكعّب بالسنتمترات المربّعة، من نمط صحيح.
- type – نوع المكعّب، من نمط رمز.

افترِضوا أنّه توجد عمليّة get/Get لصفتَي الفئة.

معطاة الفئة **Box** – علبة، ولها صفتان:

- cubes – مصفوفة بِكِبَر 3 من نمط **Cube**، تُحفَظ فيها المكعّبات.
  يمكن إدخال مكعّبات إلى المصفوفة، بحيث تكون كلّ المكعّبات من نفس النوع أو من نفس الكِبَر.
- count – كمّيّة المكعّبات في المصفوفة.

أمامكم عمليّتا الفئة Box. بإمكانكم استعمال العمليّتين بدون حاجة للتطبيق:

الشرح	العمليّة

عمليّة بنائيّة	public Box(){
	cubes= new Cube[3];
	count = 0;
	}

تتلقّى العمليّة كائنًا من نوع Cube. إذا كان يمكن إدخال الكائن إلى المصفوفة (وفق المتطلّبات أعلاه)، تدخله العمليّة إلى المكان الأوّل الفارغ في المصفوفة (وتُحدِّث قيمة المتغيّر count) وتُعيد true. خلاف ذلك (إذا كانت المصفوفة مملوءة أو إذا كان الكائن لا يفي بالمتطلّبات أعلاه)، تُعيد العمليّة false بدون تغيير قيَم المصفوفة.	**Java**: public boolean add(Cube c) **C#**: public bool Add(Cube c)

أ. تُسمّى العلبة "مثاليّة" إذا كان فيها 3 مكعّبات من نفس النوع وأيضًا من نفس الكِبَر.
طبِّقوا العمليّة الداخليّة التي أمامكم في الفئة Box:

Java: public boolean isPerfect ()
C#: public bool IsPerfect ()

تُعيد العمليّة true إذا كانت العلبة مثاليّة. خلاف ذلك تُعيد false.

ب. في مصنع ألعاب، يُعبّئون مكعّبات في العُلَب حسب ترتيب وصولها.
(1) طبِّقوا العمليّة الخارجيّة التي أمامكم:

Java: public static Box packBox (Queue<Cube> q)
C#: public static Box PackBox (Queue<Cube> q)

تتلقّى العمليّة دَوْرًا لمكعّبات، وتُعيد كائنًا من نوع Box. تُدخِل العمليّة مكعّبات من الدَّوْر (حسب ترتيب ظهورها) إلى العلبة، طالما كانت عمليّة الإدخال ممكنة. إذا لم يكن من الممكن إدخال المكعّب التالي إلى الدَّوْر، تتوقّف العمليّة، وتُعيد العلبة التي تكوَّنت.
ملاحظة: في نهاية العمليّة، ترتيب المكعّبات في الدَّوْر الأصليّ يُحفَظ، بدون المكعّبات التي أُدخلت إلى العلبة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 5 /

مثال: بالنسبة للدَّوْر الذي أمامكم:

نهاية الدَّوْر رأس الدَّوْر

Cube	Cube	Cube	Cube	Cube	Cube	Cube	Cube
size= 1 type= 'a'	size= 2 type= 'a'	size= 2 type= 'b'	size= 1 type= 'a'	size= 1 type= 'a'	size= 1 type= 'a'	size= 3 type= 'b'	size= 3 type= 'a'

تُعيد العمليّة كائنًا من نوع Box ، فيه قيمة count هي 2 والمصفوفة cubes هي:

Cube	Cube	null
size= 1 type= 'a'	size= 2 type= 'a'	

حالة الدَّوْر في نهاية العمليّة هي:

نهاية الدَّوْر رأس الدَّوْر

Cube	Cube	Cube	Cube	Cube	Cube
size= 2 type= 'b'	size= 1 type= 'a'	size= 1 type= 'a'	size= 1 type= 'a'	size= 3 type= 'b'	size= 3 type= 'a'

الشرح: نوع المكعَّبين الأوّلين متطابق، لذلك يمكن إدخالهما إلى العلبة. المكعّب الثالث من نوع مختلف، لذلك لا يدخل إلى العلبة.

(2) اكتبوا عمليّة خارجيّة باسم packAll بلغة Java أو PackAll بلغة C# ، تتلقَّى دَوْرًا q لكائنات من نوع Cube. العمليّة:

- تُعيد دَوْرًا من نوع Box ، بعد إدخال جميع المكعّبات إلى العُلَب، حسب القواعد المعطاة أعلاه.
- تطبع perfect بعد إدخال جميع المكعّبات إلى العُلَب، إذا كانت توجد، على الأقلّ، علبة واحدة "مثاليّة"، خلاف ذلك تطبع العمليّة not perfect.

مثال: بالنسبة للدَّوْر من البند "أ"، تُعيد العمليّة الدَّوْر الذي أمامكم وتطبع perfect.

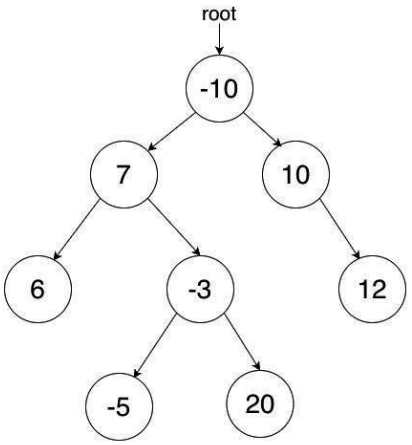
نهاية الدَّوْر رأس الدَّوْر

Box			Box			Box			Box		
Cubes			Cubes			Cubes			Cubes		
size= 1 type= 'a'	size= 2 type= 'a'	null	size= 2 type= 'b'	null	null	size= 1 type= 'a'	size= 1 type= 'a'	size= 1 type= 'a'	size= 3 type= 'b'	size= 3 type= 'a'	null
count = 2			count = 1			count = 3			count = 2		

الشرح: وفق المتطلّبات الواردة أعلاه، توجد حاجة إلى أربع عُلَب، العلبة الثالثة في الدَّوْر هي علبة مثاليّة.

/ يتبع في صفحة 6 /

3. أمامكم شجرة بينارّية، جذرها root يحوي أعدادًا صحيحة:



أ. معطاة العمليّة something بلغة **Java** وَ Something بلغة **C#**:

Java	public static int something (BinNode<Integer> root) {
	if (root == null)
	return 0;
	if (root.getLeft() == null && root.getRight() == null)
	return root.getValue ();
	int ans1 = something(root.getLeft());
	int ans2 = something(root.getRight());
	if (ans1 > ans2)
	return ans1 + root.getValue();
	return ans2 + root. getValue();
	}

C#	public static int Something (BinNode<int> root) {
	if (root == null)
	return 0;
	if (root.GetLeft() == null && root.GetRight ()== null)
	return root.GetValue();
	int ans1 = Something(root.GetLeft());
	int ans2 = Something(root.GetRight());
	if (ans1 > ans2)
	return ans1 + root.GetValue();
	return ans2 + root. GetValue();
	}

(1) ما هي القيمة التي تُعيدها العمليّة بالنسبة للاستدعاء: somthing (root) بلغة Java أو Somthing (root) بلغة #C مع الشجرة المعطاة؟ يجب عرض متابعة لتنفيذ العمليّة.

(2) اشرحوا ماذا تُنفِّذ العمليّة بالنسبة لكلّ شجرة لأعداد صحيحة.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

ب. معطاة العمليّة sod بلغة **Java** وَ Sod بلغة **C#** ، تتلقّى شجرة بينارِيّة لأعداد صحيحة، في مجال القيَم 100- حتّى 100:

Java

```
public static int sod (BinNode<Integer> root) {
   if (root == null)
         return -101;
   int x = something (root);
   int left = sod (root.getLeft());
   int right = sod (root.getRight());
   return Math.max (x, Math.max (left, right));
}
```

C#

```
public static int Sod (BinNode<int> root) {
   if (root == null)
         return -101;
   int x = Something (root);
   int left = Sod (root.GetLeft());
   int right = Sod (root.GetRight());
   return Math.Max (x, Math.Max (left, right));
}
```

ما هي القيمة التي تُعيدها العمليّة بالنسبة للاستدعاء: sod (root) بلغة Java أو Sod (root) بلغة C# مع الشجرة المعطاة أعلاه؟ يجب عرض متابعة لتنفيذ العمليّة.

لا حاجة لتنفيذ متابعة للعمليّة Something/ something ، وإنّما فقط ذِكْر القيمة التي أعيدت من العمليّة.

/ يتبع في صفحة 8 /

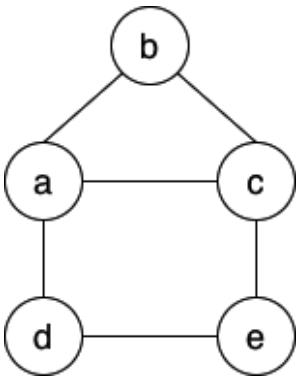
الفصل الثاني (50 درجة)

في هذا الفصل أسئلة في ثلاثة مسارات:
ألغوريثمات، الأسئلة 4-6.
موديلات حسابيّة، الأسئلة 7-9.
برمجة موجّهة كائنات (بلغة Java وبلغة C#)، الأسئلة 10-12.
يجب الإجابة عن أسئلة من مسار واحد فقط.

ألغوريثمات

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

أ. أمامكم رسم بيانيّ ليس موجّهًا G = (V, E).



(1) هل الرسم البيانيّ قابل للرّبْط؟ علِّلوا.

(2) هل الرسم البيانيّ يحوي دائرة؟ إذا كانت الإجابة نعم، اذكروا دائرة واحدة في الرسم البيانيّ. إذا كانت الإجابة لا، علِّلوا.

(3) إذا كان الرسم البيانيّ ذا جانبين، اعرضوا تقسيمًا ممكنًا (اذكروا العُقَد في كلّ جانب).
إذا لم يكن الرسم البيانيّ ذا جانبين، أزيلوا أقلّ عدد ممكن من الأقواس بحيث يكون الرسم البيانيّ الذي ينتج ذا جانبين، واعرضوا تقسيمًا ممكنًا لمجموعتين.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

ب. معطى رسم بيانيّ ليس موجّهًا G = (V, E) تُمثّله مصفوفة المجاوَرات التالية:

	a	**b**	**c**	**d**	**e**	**f**
a	0	1	1	0	0	0
b	1	0	0	1	0	0
c	1	0	0	1	1	0
d	0	1	1	0	0	0
e	0	0	1	0	0	1
f	0	0	0	0	1	0

(مفتاح: 1 – يوجد قوس، 0 – لا يوجد قوس.)

(1) ارسموا الرسم البيانيّ الذي تُمثّله المصفوفة المعطاة.

(2) كم مركِّب رَبْط يوجد في الرسم البيانيّ G ؟ علِّلوا.

(3) اعرضوا أقصر مسار من الرأس a إلى الرأس f .

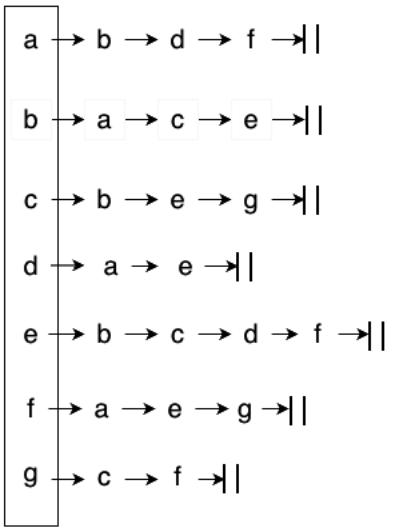
(4) هل يوجد قوس من الرسم البيانيّ G ، تزيد إزالته عدد مركِّبات الرَّبْط بِـ 1؟ إذا كانت الإجابة نعم، اذكروا القوس.

(5) هل يوجد قوس من الرسم البيانيّ G ، لا تُغيِّر إزالته عدد مركِّبات الرَّبْط في الرسم البيانيّ؟ إذا كانت الإجابة نعم، اذكروا القوس.

/ يتبع في صفحة 10 /

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

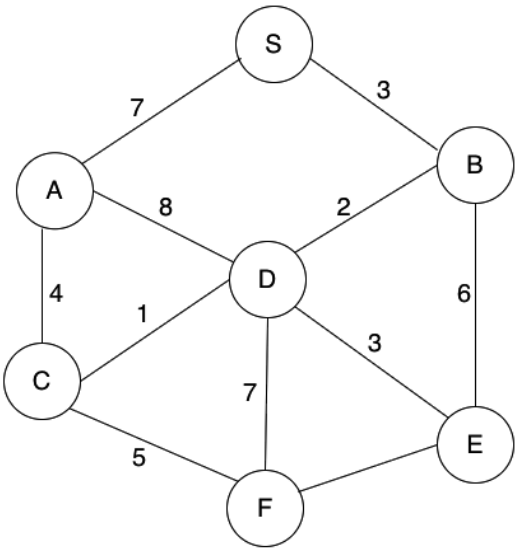
أ. معطى رسم بيانيّ ليس موجّهًا $G = (V, E)$ ، تُمثِّله قائمة المجاوَرات التي أمامكم.



(1) شغِّلوا ألغوريثم مسح عميق DFS على الرسم البيانيّ المعطى، ابتداءً من الرأس a. ارسموا في دفتركم شجرة المسح DFS التي تنتج.

(2) شغِّلوا ألغوريثم مسح عريض BFS على الرسم البيانيّ المعطى، ابتداءً من الرأس a. ارسموا في دفتركم شجرة المسح BFS التي تنتج.

ب. أمامكم رسم بيانيّ موزون ليس موجَّهًا G . لكلّ قوس يوجد وزن موجب.



(1) اكتبوا أقصر مسارات (الأخفّ وزنًا) في الرسم البيانيّ G من الرأس S إلى كلّ واحد من الرؤوس في الرسم البيانيّ. لا حاجة لعرض متابعة.

(2) بالنسبة للرسم البيانيّ G المعطى، ارسموا شجرة المسارات الأقصر من الرأس S.

/ يتبع في صفحة 11 /

6. في هذا السؤال 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 لم يكن صحيحًا. أعطوا مثالًا مناقضًا من رسم بيانيّ فيه أربعة رؤوس على الأقلّ.

(1) في رسم بيانيّ ليس موجَّهًا فيه لكلّ رأس درجة زوجيّة أكبر من 0 ، توجد دائرة.

(2) في رسم بيانيّ ليس موجَّهًا قابل للرَّبْط، إزالة قوس لا ينتمي إلى أيّة دائرة، تجعل الرسم البيانيّ ليس قابلًا للرَّبْط.

(3) كلّ شجرة هي رسم بيانيّ ذو جانبين، وأيضًا كلّ رسم بيانيّ ذو جانبين هو شجرة.

(4) إذا كانت هناك دائرة في رسم بيانيّ موجَّه، إذًا الرسم البيانيّ قابل للرَّبْط جيّدًا (قويّ).

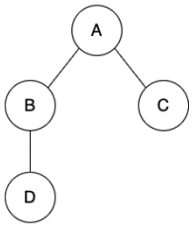
(5) إذا كان هناك رسم بيانيّ ليس موجَّهًا وهو شجرة، إذًا بين كلّ رأسين في الرسم البيانيّ يوجد مسار وحيد.

(6) إذا أضفنا قوسًا واحدًا أيًّا كان إلى رسم بيانيّ ليس موجَّهًا بدون دوائر، فبالضرورة تتكوّن دائرة.

(7) معطى رسم بيانيّ ليس موجَّهًا فيه n رؤوس. إذا كان يوجد في الرسم البيانيّ n-1 أقواس، فبالضرورة هو شجرة.

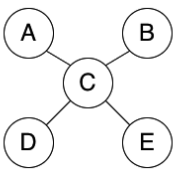
ب. شجرة تُسمّى "**شجرة ضحلة**" إذا كان من كلّ عقدة في الشجرة يوجد بُعد قدره 3 أقواس كحدّ أقصى عن كلّ عقدة أخرى في الشجرة.

مثال لِـ "شجرة ضحلة":

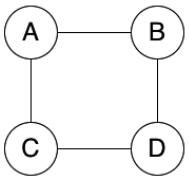


رسم بيانيّ ليس موجَّهًا وقابل للرَّبْط يُسمّى "**رسم بيانيّ BFS ضحل**" إذا كان في كلّ تشغيل BFS على الرسم البيانيّ، من كلّ رأس بداية اختير، تنتج "شجرة ضحلة".

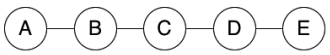
أمامكم ستّة رسوم بيانيّة G1 – G6. اختاروا أربعة منها، واكتبوا بالنسبة لكلّ رسم بيانيّ اخترتموه هل هو "رسم بيانيّ BFS ضحل" أم لا. إذا كانت الإجابة لا – اعرضوا مسح BFS تنتج بالنسبة له شجرة ليست "ضحلة".



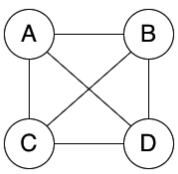
G1



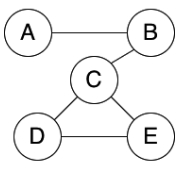
G2



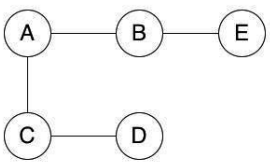
G3



G4



G5



G6

/ يتبع في صفحة 12 /

موديلات حسابيّة

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

أ. أمامكم اللغتان النظاميّتان L_1 وَ L_2 فوق الأبجديّة $\{a, b\}$:

$L_1 = \{a^{2k} \mid k>=0\}$

$L_2 = \{b^{2k+1} \mid k>=0\}$

برهِنوا بالاستعانة بصفات انغلاق اللغات النظاميّة أنّ اللغة L_3 نظاميّة.

$L_3 = \{a^n b^m \mid n,m>=0,\ m\%2 \neq n\%2\}$

ب. معطاة اللُّغتان التاليتان:

$L = \{a^z w_1 w_2 \ldots w_k d^{k+1} \mid z\%2=1,\ w_i \in L_1,\ k>0\}$

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

مثال لكلمة تتبع للُّغة L : aaabbcbccddd

a^z	w_1	w_k	d^{k+1}
aaa	bbc	bcc	ddd

الشرح: عدد رموز a هو فرديّ. بين رموز a في بداية الكلمة ورموز d في آخر الكلمة، توجد كلمتان تتبعان للُّغة L_1 ، لذلك k=2 ، ووفقًا لذلك عدد رموز d في نهاية الكلمة هو 3.

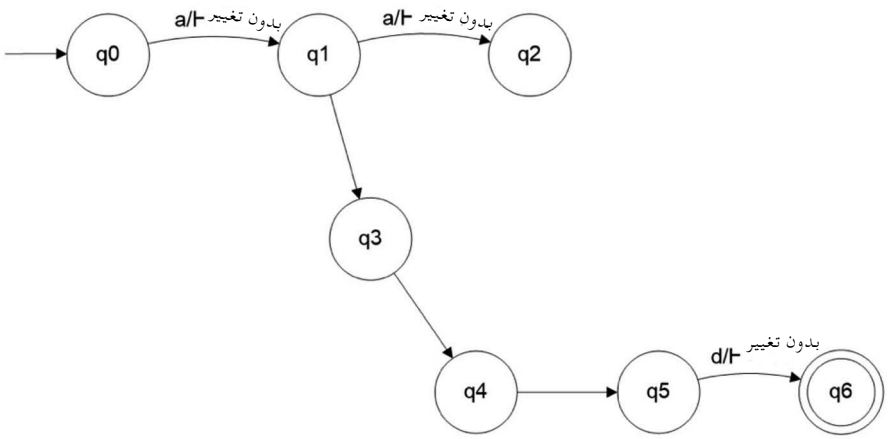
(1) اكتبوا أقصر كلمة في اللغة L.

(2) معطى أوتومات راصّة محدود جزئيّ يتلقّى اللغة L. الأوتومات يحوي جميع الحالات (بما في ذلك إشارة إلى الحالة المتلقّية).

عليكم إكمال الانتقالات الناقصة وتفصيل الانتقالات القائمة (الرمز في الانتقال والإشارة في رأس الراصّة والعمليّة على الراصّة).

ملاحظة: لا تضيفوا أو تزيلوا حالات من الأوتومات.

انسخوا أوتومات الراصّة إلى دفتر الامتحان، وأكمِلوه بحيث يتلقّى اللغة L.



/ يتبع في صفحة 13 /

8. معطاة اللُّغات L_1 وَ L_2 وَ L_3 فوق الأبجديّة $\{a, b\}$:

$L_1=\{w\cdot R(w) \mid w\in\{a^n b^m \mid n=m\%2,\ m\geq 0\}\}$

$L_2=\{a^n b^{2n} a^n \mid n\geq 0\}$

$L_3=\{(ab)^n (ba)^m \mid n\geq m,\ m\geq 0\}$

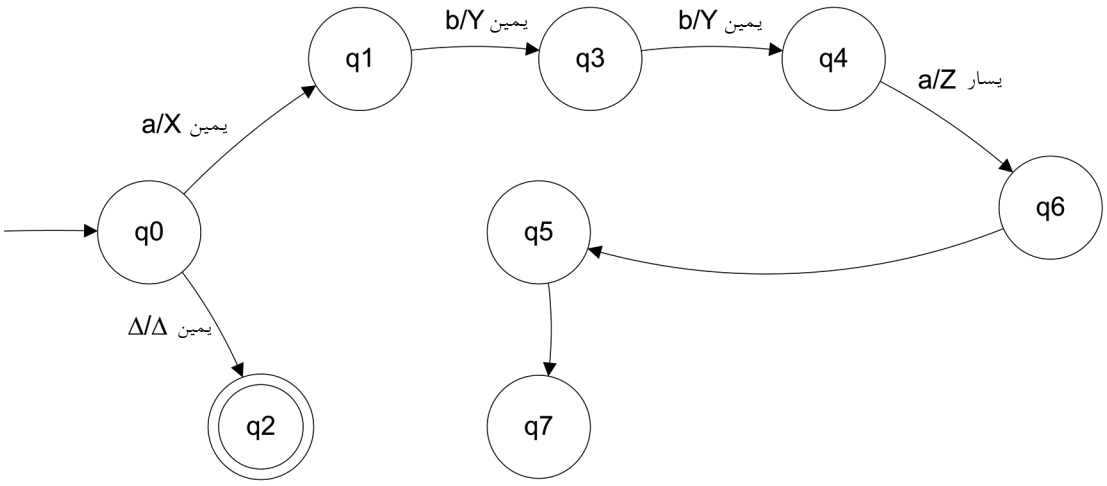
أ. **(1)** اكتبوا بالنسبة لكلّ لغة، هل هي نظاميّة أم لا، وعلِّلوا باختصار.

(2) اكتبوا كلمة ليست فارغة تتبع للغة التقاطُع $L_1 \cap L_3$.

ب. أمامكم آلة تيورينج جزئيّة تفحص هل الكلمة في بداية الشريط تتبع للُّغة L_2 (المعطاة أعلاه).
الآلة تحوي جميع الحالات اللازمة (بما في ذلك إشارة إلى الحالة المتلقّية).
انسخوا إلى دفتر الامتحان الآلة الجزئيّة المعطاة، وأكمِلوها بحيث تتلقّى اللغة L_2 .

ملاحظات:

- لا حاجة لحفظ كلمة الإدخال على الشريط.
- لا تضيفوا حالات.
- يمكن إضافة انتقالات.



/ يتبع في صفحة 14 /

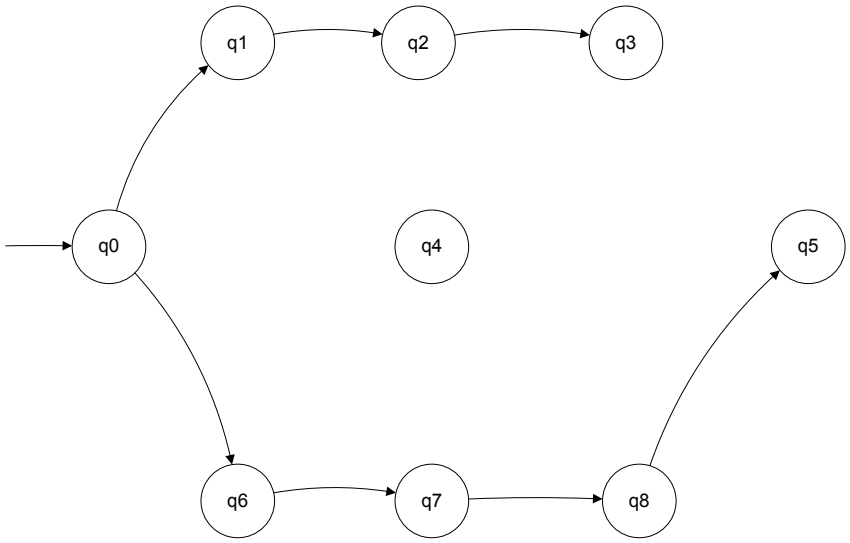
9. في هذا السؤال بندان، "أ – ب"، لا علاقة بينهما. أجيبوا عن البندين.

أ. معطاة اللّغة L فوق الأبجديّة {a, b}، تحوي الكلمات التي تُحقِّق جميع الشروط التي أمامكم:

- الكلمة لا تبدأ بِـ ab.
- إذا كانت الكلمة تحوي aa ، إذًا هي لا تحوي bb.
- إذا كانت الكلمة تحوي bb ، إذًا هي لا تحوي aa.
- يجب أن تحوي الكلمة aa أو bb مرّة واحدة، على الأقلّ.

أمامكم أوتومات نهائيّ محدود كامل يتلقّى اللغة L . ينقص في الأوتومات انتقالات وإشارات إدخال وحالات متلقّية. انسخوا الأوتومات إلى دفتر الامتحان، وأكمِلوه بحيث يتلقّى اللغة L.

ملاحظة: لا تضيفوا أو تزيلوا حالات، ولا تزيلوا انتقالات من الأوتومات المعطى.



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

$$L=\{a^{2m}b^{m-1}c \mid m\geq 1\}$$

(1) ما هي أقصر كلمة في اللغة L ؟

(2) هل اللغة L هي نظاميّة؟ علِّلوا باختصار.

/ يتبع في صفحة 15 /

برمجة موجّهة كائنات

10. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 15-16، وبلغة **C#** في الصفحتين 17-18.

للّذين يحلُّون بلغة Java

أمامكم عنوانا الفئتين BB ، CC ، وصفاتهما:

```
public class BB
{
  private int a;
  protected int b;

    ****
}

public class CC extends BB
{
  private int c;

    ****
}
```

ملاحظة: انتبهوا – لا توجد عمليّات get وَ set للفئتين.

أمامكم الفئة Tester التي تحوي عمليّة رئيسيّة:

```
public class Tester{
  public static void main(String[] args)
  {
    BB [] b = new BB[8];
    b[0] = new BB();
    b[1] = new BB(16);
    b[2] = new BB(b[0]);
    b[3] = new BB(b[1]);
    b[4] = new CC(9);
    b[5] = new CC(5);
    b[6] = new CC(b[0]);
    b[7] = new CC(b[1]);
  }
}
```

أ. أمامكم مخطَّط الكائنات التي تكوَّنت في أعقاب تشغيل قطعة الكود.

اكتبوا في الفئتين BB وَ CC العمليّات البنائيّة اللازمة للحصول على الكائنات التي في المخطَّط.

اذكروا بالنسبة لكلّ واحد من الكائنات التي في المخطَّط، العمليّة البنائيّة التي تكوّن هذا الكائن حسبها.

ملاحظة: لا تضيفوا عمليّات ليست بنائيّة في الفئتين BB وَ CC. الحلّ الذي يتضمّن عمليّات ليست بنائيّة لن يحصل على درجات.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 16 /

0	1	2	3	4	5	6	7
BB	BB	BB	BB	CC	CC	CC	CC
a = 10 b = 1	a = 10 b = 16	a = 10 b = 1	a = 10 b = 16	a = 10 b = 9 c = 7	a = 10 b = 5 c = 7	a = 10 b = 1 c = 19	a = 10 b = 16 c = 19

ب. معطاة قطعة كود سليمة والمُخرَج الذي نتج في أعقاب تشغيلها. وكذلك، معطاة قطعة كود نتج بالنسبة لها خطأ برمجيّ. طبِّقوا العمليّتين goo وَ foo في الفئتين الملائمتين (BB وَ CC) حسب مبادئ البرمجة الموجّهة كائنات، بحيث ينتج المُخرَج الذي أمامكم (يجب أن تطبِّقوا فقط العمليّات الضروريّة في الفئتين حتّى يُشغَّل الكود).

قطعة كود سليمة:

```
b[0].foo(0);          //1
b[0].foo(3);          //2
((CC)b[4]).foo();     //3
((CC)b[6]).foo();     //4
((CC)b[5]).goo();     //5
((CC)b[7]).goo();     //6
```

المُخرَج الذي نتج بالنسبة لتشغيله:

```
BB foo: 11          //1
BB foo: 14          //2
CC foo: 7           //3
BB foo: 26
CC foo: 19          //4
BB foo: 30
CC goo              //5
CC goo              //6
```

إضافة إلى ذلك، العمليّتان في قطعة الكود الذي أمامكم ليستا سليمتين، وينتج خطأ برمجيّ.

```
b[3].foo();
b[2].goo();
```

/ يتبع في صفحة 17 /

للّذين يحلُّون بلغة #C

أمامكم عنوانا الفئتين BB ، CC ، وصفاتهما :

```
public class BB
{
   private int a;
   protected int b;

      ****
}

public class CC: BB
{
   private int c;

      ****
}
```

ملاحظة: انتبهوا – لا توجد عمليّات Get وَ Set للفئتين.

أمامكم الفئة Tester التي تحوي عمليّة رئيسيّة :

```
public class Tester{
   public static void Main(string[] args)
   {
      BB [] b = new BB[8];
      b[0] = new BB();
      b[1] = new BB(16);
      b[2] = new BB(b[0]);
      b[3] = new BB(b[1]);
      b[4] = new CC(9);
      b[5] = new CC(5);
      b[6] = new CC(b[0]);
      b[7] = new CC(b[1]);
   }
}
```

أ. أمامكم مخطَّط الكائنات التي تكوَّنت في أعقاب تشغيل قطعة الكود.

اكتبوا في الفئتين BB وَ CC العمليّات البنائيّة اللازمة للحصول على الكائنات التي في المخطَّط.

اذكروا بالنسبة لكلّ واحد من الكائنات التي في المخطَّط، العمليّة البنائيّة التي تكوّن هذا الكائن حسبها.

ملاحظة: لا تضيفوا عمليّات ليست بنائيّة في الفئتين BB وَ CC. الحلّ الذي يتضمّن عمليّات ليست بنائيّة لن يحصل على درجات.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 18 /

0	1	2	3	4	5	6	7
BB	BB	BB	BB	CC	CC	CC	CC
a = 10 b = 1	a = 10 b = 16	a = 10 b = 1	a = 10 b = 16	a = 10 b = 9 c = 7	a = 10 b = 5 c = 7	a = 10 b = 1 c = 19	a = 10 b = 16 c = 19

ب. معطاة قطعة كود سليمة والمُخرَج الذي نتج في أعقاب تشغيلها. وكذلك، معطاة قطعة كود نتج بالنسبة لها خطأ برمجيّ. طبِّقوا العمليّتين Goo وَ Foo في الفئتين الملائمتين (BB وَ CC) حسب مبادئ البرمجة الموجّهة كائنات، بحيث ينتج المُخرَج الذي أمامكم (يجب أن تطبِّقوا فقط العمليّات الضروريّة في الفئتين حتّى يُشغَّل الكود).

قطعة كود سليمة:

```
b[0].Foo(0);          //1
b[0].Foo(3);          //2
((CC)b[4]).Foo();     //3
((CC)b[6]).Foo();     //4
((CC)b[5]).Goo();     //5
((CC)b[7]).Goo();     //6
```

المُخرَج الذي نتج بالنسبة لتشغيله:

```
BB Foo: 11          //1
BB Foo: 14          //2
CC Foo: 7           //3
BB Foo: 26
CC Foo: 19          //4
BB Foo: 30
CC Goo              //5
CC Goo              //6
```

إضافة إلى ذلك، العمليّتان في قطعة الكود الذي أمامكم ليستا سليمتين، وينتج خطأ برمجيّ.

```
b[3].Foo();
b[2].Goo();
```

/ يتبع في صفحة 19 /

11. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 19-20، وبلغة **C#** في الصفحتين 21–22.

<u>للّذين يحلُّون بلغة Java</u>

بنوا في محلّ ألعاب منظومة لإدارة الألعاب، فيها الفئات التالية:
Game , VRGame , ComputerGame , BoardGame .
أمامكم تفصيل صفات الفئات:

- **Game** – لعبة
  صفات الفئة:
  - rate – تدريج، من نمط عدد صحيح
  - code – كود اللعبة، من نمط صحيح
  - price – سعر اللعبة، من نمط صحيح
- **ComputerGame** – لعبة حاسوب
  صفات الفئة:
  - rate – تدريج، من نمط عدد صحيح
  - code – كود اللعبة، من نمط صحيح
  - price – سعر اللعبة، من نمط صحيح
  - memory – حدّ أدنى للذاكرة، من نمط صحيح
- **VRGame** – لعبة واقع افتراضيّ
  صفات الفئة:
  - rate – تدريج، من نمط عدد صحيح
  - code – كود اللعبة، من نمط صحيح
  - price – سعر اللعبة، من نمط صحيح
  - memory – حدّ أدنى للذاكرة، من نمط صحيح
  - needVR – هل تلزم نظّارات، من نمط بوليانيّ
  - age – عمر أدنى للّاعب في اللعبة، من نمط صحيح
- **BoardGame** – لعبة لوحيّة
  صفات الفئة:
  - rate – تدريج، من نمط عدد صحيح
  - code – كود اللعبة، من نمط صحيح
  - price – سعر اللعبة، من نمط صحيح
  - players أكبر عدد للّاعبين في اللعبة، من نمط صحيح

أ. **(1)** ارسموا مخطَّطًا هرميًّا يصف العلاقة بين فئات المنظومة المُحَوْسَبة.
يجب الإشارة إلى توريث بواسطة السهم ⟵ .

(2) اكتبوا عناوين الفئات وصفاتها حسب مبادئ البرمجة الموجّهة كائنات. افترِضوا أنّ عمليّات get وَ set موجودة لجميع صفات الفئات، ولا حاجة لتطبيقها.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

معطى عنوان العمليّة البنائيّة للفئة Game:

public Game(int price, int code)

لا حاجة لتطبيق العمليّة. تقييم اللعبة مبتدَأ إلى القيمة 0.

(3) أمامكم عنوان العمليّة البنائيّة للفئة ComputerGame. تتلقّى العمليّة سعرًا price،
وكودًا code ، وحدًّا أدنى للذاكرة memory .

public ComputerGame (int price, int code, int memory)

طبِّقوا العمليّة البنائيّة.

ب. معطاة الفئة Store – محلّ، ولها صفتان:

- games – مصفوفة ألعاب، من نمط Game بكِبَر 1000.
- count – كمّيّة الألعاب في المصفوفة.

تُحفَظ الألعاب من بداية المصفوفة بتسلسل، لكن يمكن ألّا تكون المصفوفة مملوءة حتّى النهاية. إضافةً إلى ذلك، المصفوفة ليست مرتّبة.

اكتبوا عمليّة داخليّة باسم avgRate في الفئة Store ، تتلقّى عددًا يُمثِّل نوع اللعبة:
العدد 1 – يُمثِّل اللعبة VRGame.
العدد 2 – يُمثِّل اللعبة ComputerGame.
تُعيد اللعبة عددًا حقيقيًّا يُمثِّل معدّل تدريجات جميع الألعاب من نفس النوع في المحلّ.
افترِضوا أنّه توجد في المصفوفة لعبة واحدة، على الأقلّ، من كلّ نوع.

/ يتبع في صفحة 21 /

<u>للّذين يحلُّون بلغة #C</u>

بنوا في محلّ ألعاب منظومة لإدارة الألعاب، فيها الفئات التالية:
Game , VRGame , ComputerGame , BoardGame .
أمامكم تفصيل صفات الفئات:

- **Game** – لعبة

صفات الفئة:

- rate – تدريج، من نمط عدد صحيح
- code – كود اللعبة، من نمط صحيح
- price – سعر اللعبة، من نمط صحيح

- **ComputerGame** – لعبة حاسوب

صفات الفئة:

- rate – تدريج، من نمط عدد صحيح
- code – كود اللعبة، من نمط صحيح
- price – سعر اللعبة، من نمط صحيح
- memory – حدّ أدنى للذاكرة، من نمط صحيح

- **VRGame** – لعبة واقع افتراضيّ

صفات الفئة:

- rate – تدريج، من نمط عدد صحيح
- code – كود اللعبة، من نمط صحيح
- price – سعر اللعبة، من نمط صحيح
- memory – حدّ أدنى للذاكرة، من نمط صحيح
- needVR – هل تلزم نظّارات، من نمط بوليانيّ
- age – عمر أدنى للّاعب في اللعبة، من نمط صحيح

- **BoardGame** – لعبة لوحيّة

صفات الفئة:

- rate – تدريج، من نمط عدد صحيح
- code – كود اللعبة، من نمط صحيح
- price – سعر اللعبة، من نمط صحيح
- players أكبر عدد للّاعبين في اللعبة، من نمط صحيح

أ. **(1)** ارسموا مخطَّطًا هرميًّا يصف العلاقة بين فئات المنظومة المُحَوْسَبة.
يجب الإشارة إلى توريث بواسطة السهم ⟵ .

(2) اكتبوا عناوين الفئات وصفاتها حسب مبادئ البرمجة الموجّهة كائنات. افترِضوا أنّ عمليّات Get وَ Set موجودة لجميع صفات الفئات، ولا حاجة لتطبيقها.

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 22 /

معطى عنوان العمليّة البنائيّة للفئة Game:

public Game(int price, int code)

لا حاجة لتطبيق العمليّة. تقييم اللعبة مبتدَأ إلى القيمة 0.

(3) أمامكم عنوان العمليّة البنائيّة للفئة ComputerGame. تتلقّى العمليّة سعرًا price،
وكودًا code ، وحدًّا أدنى للذاكرة memory .

public ComputerGame (int price, int code, int memory)

طبِّقوا العمليّة البنائيّة.

ب. معطاة الفئة Store – محلّ، ولها صفتان:

- games – مصفوفة ألعاب، من نمط Game بكِبَر 1000.
- count – كمّيّة الألعاب في المصفوفة.

تُحفَظ الألعاب من بداية المصفوفة بتسلسل، لكن يمكن ألّا تكون المصفوفة مملوءة حتّى النهاية. إضافةً إلى ذلك، المصفوفة ليست مرتّبة.

اكتبوا عمليّة داخليّة باسم AvgRate في الفئة Store ، تتلقّى عددًا يُمثِّل نوع اللعبة:
العدد 1 – يُمثِّل اللعبة VRGame.
العدد 2 – يُمثِّل اللعبة ComputerGame.
تُعيد اللعبة عددًا حقيقيًّا يُمثِّل معدّل تدريجات جميع الألعاب من نفس النوع في المحلّ.
افترِضوا أنّه توجد في المصفوفة لعبة واحدة، على الأقلّ، من كلّ نوع.

/ يتبع في صفحة 23 /

12. لهذا السؤال صيغتان: بلغة **Java** في الصفحتين 23-24، وبلغة **C#** في الصفحتين 25–26.

<u>للّذين يحلُّون بلغة Java</u>

أمامكم الفئات E, F, G, H :

```
public class E {
   protected int x;
   public E() {
      this.x = 5;
      System.out.println("E x = " + this.x);
   }
   public E(int x) {
      this.x = x;
   }
   public int calc() {
      return this.x;
   }
}
-----------------------------------------------
public class F extends E {
   private static int count = 0;
   public F() {
      super(8);
      count++;
   }
   public F(int x) {
      super(x);
      count++;
      System.out.println("F count = " + count);
   }
   public int calc() {
      return this.x + 10;
   }
}
```

```
public class G extends F {
   public G() {
      super(3);
      this.x =this.x* 2;
      System.out.println("G x = " + this.x);
   }
   public int calc() {
      return super.calc() + 1;
   }
   public int squre() {
      return this.x*this.x;
   }
}
-----------------------------------------------
public class H extends E {
   public H() {
      super();
      this.x = this.x + 5;
      System.out.println("H x = " + this.x);
   }
   public int calc() {
      return this.x/2;
   }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 24 /

معطاة الفئة Tester :

```
public class Tester
{
  public static void main(String[] args)
  {
    E e1 = new E();
    E e2 = new F();
    F f1 = new G();
    E e3 = new H();
    E a = new G();
    E b = new H();
    F c = new G();
    // ***
  }
}
```

أ. اعرضوا الكائنات التي تكوّنت في العمليّة main (نوع الكائن ونوع التوجيه وقيَم الصفات)، واكتبوا مُخرَج العمليّة.

ب. عوِّضوا كلّ واحد من الأوامر 1–10 التالية في العمليّة main في المكان المُشار إليه أعلاه بِـ *** // .
اكتبوا في الدفتر رقم الأمر، واذكروا هل الكود الناتج سليم أم غير سليم.
إذا كان الكود سليمًا – اكتبوا المُخرَج. إذا لم يكن الكود سليمًا، اشرحوا لماذا.

انتبهوا: لا علاقة بين الأوامر التالية. الأوامر لا يتعلّق أحدها بالآخر.

1. `F f = new E();`
2. `G g = new F();`
3. `F f2 = (F)a;`
4. `G g2 = (G)c;`
5. `F f3 = (F)b;`
6. `System.out.println(((G)a).squre());`
7. `System.out.println(a.squre());`
8. `System.out.println(((H)b).calc());`
9. `System.out.println(((G)b).squre());`
10. `System.out.println(f1.squre());`

/ يتبع في صفحة 25 /

للّذين يحلُّون بلغة #C

أمامكم الفئات E, F, G, H :

```
public class E {
   protected int x;
   public E(){
      this.x = 5;
      Console.WriteLine("E x = " + this.x);
   }
   public E(int x) {
      this.x = x;
   }
   public int Calc() {
      return this.x;
   }
}
-----------------------------------------------
public class F: E {
   private static int count = 0;
   public F(): base(8){
      count++;
   }
   public F(int x): base(x) {
      count++;
      Console.WriteLine("F count = " + count);
   }
   public int Calc() {
      return this.x + 10;
   }
}
```

```
public class G: F {
   public G(): base(3) {
      this.x = this.x * 2;
      Console.WriteLine("G x = " + this.x);
   }
   public int Calc() {
      return base.Calc() + 1;
   }
   public int Squre() {
      return this.x*this.x;
   }
}
-----------------------------------------------
public class H: E {
   public H(): base(){
      this.x = this.x + 5;
      Console.WriteLine ("H x = " + this.x);
   }
   public int Calc() {
      return this.x/2;
   }
}
```

(انتبهوا: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 26 /

معطاة الفئة Tester:

```
public class Tester
{
  public static void Main(string[] args)
  {
    E e1 = new E();
    E e2 = new F();
    F f1 = new G();
    E e3 = new H();
    E a = new G();
    E b = new H();
    F c = new G();

    // ***
  }
}
```

أ. اعرضوا الكائنات التي تكوّنت في العمليّة Main (نوع الكائن ونوع التوجيه وقيَم الصفات)، واكتبوا مُخرَج العمليّة.

ب. عوِّضوا كلّ واحد من الأوامر 1–10 التالية في العمليّة Main في المكان المُشار إليه أعلاه بِـ *** // .
اكتبوا في الدفتر رقم الأمر، واذكروا هل الكود الناتج سليم أم غير سليم.
إذا كان الكود سليمًا – اكتبوا المُخرَج. إذا لم يكن الكود سليمًا، اشرحوا لماذا.

انتبهوا: لا علاقة بين الأوامر التالية. الأوامر لا يتعلّق أحدها بالآخر.

1. `F f = new E();`
2. `G g = new F();`
3. `F f2 = (F)a;`
4. `G g2 = (G)c;`
5. `F f3 = (F)b;`
6. `Console.WriteLine(((G)a).Squre());`
7. `Console.WriteLine(a.Squre());`
8. `Console.WriteLine(((H)b).Calc());`
9. `Console.WriteLine(((G)b).Squre());`
10. `Console.WriteLine(f1.Squre());`

ב ה צ ל ח ה!
نتمنّى لكم النّجاح!