מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"א, 2021
מספר השאלון: **899381**
תרגום לערבית (2)

מועד קיץ
موعد صيف

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: صيف 2021
رقم النّموذج: **899381**
ترجمة إلى العربيّة (2)

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

מדעי המחשב
הוראות לנבחן

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שלושה פרקים.
יש לבחור בחמש שאלות מתוך שלושת הפרקים.
לכל שאלה – 20 נקודות (20x5).
סך הכול – 100 נקודות.
שים לב: אם תבחר לענות על שאלות מן הפרק השלישי, בחר בשאלות מתוך מסלול אחד בלבד.

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

ד. הוראות מיוחדות:

1. **רשום על הכריכה החיצונית** של המחברת את ה**מסלול שלמדת**, אם תבחר לענות על שאלות מן הפרק השלישי, ואם לא – רשום **"ללא מסלול"**.
המסלול הוא אחד מארבעת המסלולים האלה: מערכות מחשב ואסמבלי, מבוא לחקר ביצועים, מודלים חישוביים, תכנות מונחה עצמים.

2. את כל התוכניות שאתה נדרש לכתוב בשפת מחשב בפרקים הראשון והשני, כתוב בשפה אחת בלבד – Java או C#.

הערה: לא יורדו לך נקודות אם תכתוב בתוכניות שאתה כותב אות גדולה במקום אות קטנה או להפך.

علم الحاسوب
تعليمات للممتحَن

أ. مدّة الامتحان: ثلاث ساعات.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج ثلاثة فصول.
عليك اختيار خمسة أسئلة من ثلاثة الفصول.
لكلّ سؤال – 20 درجة (20x5).
المجموع – 100 درجة.
انتبه: اذا اخترتَ الإجابة عن أسئلة من الفصل الثالث، اختر أسئلة من مسار واحد فقط.

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

د. تعليمات خاصّة:

1. **سجّل على الغلاف الخارجيّ** للدفتر اسم **المسار الذي تعلّمته**، إذا اخترتَ الإجابة عن أسئلة من الفصل الثالث. وإذا لم تختر – سجّل "**بدون مسار**".
المسار هو أحد المسارات الأربعة التالية: أنظمة حاسوب وأسمبلي، مقدّمة لبحث الأداءات، موديلات حسابيّة، برمجة موجّهة كائنات.

2. اكتب جميع البرامج التي يُطلب منك كتابتها بلغة حاسوب في الفصلين الأوّل والثاني، بلغة واحدة فقط – Java أو C#.

ملاحظة: لن تُخصم درجات إذا كتبتَ في البرامج التي تكتبها حرفًا كبيرًا بدلاً من حرف صغير أو بالعكس.

اكتب في دفتر الامتحان فقط. اكتب "مسوّدة" في بداية كلّ صفحة تستعملها مسوّدة.
كتابة أيّة مسوّدة على أوراق خارج دفتر الامتحان قد تسبّب إلغاء الامتحان.

التّعليمات في هذا النّموذج مكتوبة بصيغة المذكّر وموجّهة للممتحَنات وللممتحَنين على حدّ سواء.

ב ה צ ל ח ה !

نتمنّى لك النّجاح!

انتبه: الأسئلة تبدأ في صفحة 3.

الأسئلة

في هذا النموذج ثلاثة فصول.
عليك اختيار خمسة أسئلة من ثلاثة الفصول. لكلّ سؤال – 20 درجة.
سجّل على الغلاف الخارجيّ للدفتر اسم المسار الذي تعلّمته، إذا اخترتَ الإجابة عن أسئلة من الفصل الثالث. وإذا لم تختر – سجّل "بدون مسار".

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

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

الفصل الأوّل

1. اكتب عمليّة خارجيّة اسمها filter بلغة Java أو Filter بلغة C# تتلقّى مصفوفة من نمط صحيح — arr وعددًا من نمط صحيح — num. يجب على العمليّة أن تُعيد مصفوفة جديدة من نمط صحيح، تشمل فقط الأعداد التي تظهر في المصفوفة arr والتي لا تساوي num.

مثال: بالنسبة لِـ num يساوي 9 والمصفوفة arr بِكِبَر 7 التي أمامك:

	0	1	2	3	4	5	6
arr	6	**9**	2	2	**9**	4	−3

تُعيد العمليّة مصفوفة بِكِبَر 5 ، تبدو على النحو التالي:

0	1	2	3	4
6	2	2	4	−3

افترِضْ أنّه يوجد في المصفوفة arr عدد واحد على الأقلّ لا يساوي num وعدد واحد على الأقلّ يساوي num.

انتبه: كِبَر المصفوفة المُعادَة هو كعدد الحدود التي لا تساوي num في المصفوفة arr.

ملاحظة: لا توجد أهمّيّة لترتيب الأعداد في المصفوفة المُعادَة.

/ يتبع في صفحة 4 /

2. انتبه ! لهذا السؤال صيغتان : بلغة Java في الصفحتين 4-5 وبلغة #C في الصفحتين 6-7 .

للّذين يحلّون بلغة Java

معطاة الفئة **Subject** ، التي تمثّل موضوعًا في شهادة الطالب، ولها صفتان : اسم الموضوع – subName والعلامة – grade (بين 0 وَ 100) .

```
public class Subject {
    private String subName;
    private int grade;
}
```

بالإضافة إلى ذلك، معطاة الفئة **ReportCard**، التي تمثّل شهادة الطالب، ولها صفتان : اسم الطالب – stuName ومصفوفة كائنات – subArray من نمط **Subject** بِكِبَر عدد المواضع التي تعلّمها الطالب .

```
public class ReportCard {
    private String stuName;
    private Subject [] subArray;
}
```

افترِضْ أنّه توجد عمليّتا get وَ set بالنسبة لجميع الصفات في الفئتين .

أمامك واجهة تطبيق جزئيّة للفئة **ReportCard** .

عنوان العمليّة	وصف العمليّة
public ReportCard (String name, int num)	عمليّة بنائيّة تتلقّى اسم الطالب وعدد المواضيع التي تعلّمها الطالب، وتبتدىء صفات الفئة . المصفوفة subArray مُبتَدَأة لمصفوفة فارغة بِكِبَر num .
public double average ()	عمليّة تُعيد معدّل جميع العلامات في شهادة الطالب .
public boolean isExcellent ()	عمليّة تُعيد true إذا اعتُبر الطالب طالبًا متميّزًا، خلاف ذلك تُعيد العمليّة false . الطالب المتميّز : الطالب الذي معدّل علاماته هو 85 أو أعلى، وجميع علاماته هي أعلى من 54، وإحدى علاماته على الأقلّ، هي 100 .

(انتبه : تكملة السؤال في الصفحة التالية .)

/ يتبع في صفحة 5 /

أ. طبِّق العمليّة البنائيّة للفئة **ReportCard**.

ب. طبِّق العمليّة isExcellent في الفئة **ReportCard**.
يمكن استعمال العمليّة average للفئة **ReportCard** بدون تطبيقها.
افترِضْ أنّ حدود المصفوفة subArray تختلف عن null.

جـ. اكتب عمليّة خارجيّة اسمها printExcellent تتلقّى مصفوفة كائنات — array من نمط **ReportCard** ،
تمثّل طلّابًا في صفّ معيّن. تطبع العمليّة أسماء الطلّاب المتميّزين.
يجب استعمال العمليّة isExcellent للفئة **ReportCard**.
افترِضْ أنّ حدود المصفوفة array تختلف عن null.

/ يتبع في صفحة 6 /

للّذين يحلّون بلغة #C

معطاة الفئة **Subject** ، التي تمثّل موضوعًا في شهادة الطالب، ولها صفتان: اسم الموضوع – subName والعلامة – grade (بين 0 وَ 100).

```
public class Subject {
    private string subName;
    private int grade;
}
```

بالإضافة إلى ذلك، معطاة الفئة **ReportCard**، التي تمثّل شهادة الطالب، ولها صفتان: اسم الطالب – stuName ومصفوفة كائنات – subArray من نمط **Subject** بِكِبَر عدد المواضع التي تعلّمها الطالب.

```
public class ReportCard {
    private string stuName;
    private Subject [] subArray;
}
```

افترِضْ أنّه توجد عمليّتا Get وَ Set بالنسبة لجميع الصفات في الفئتين.

أمامك واجهة تطبيق جزئيّة للفئة **ReportCard** .

عنوان العمليّة	وصف العمليّة
public ReportCard (string name, int num)	عمليّة بنائيّة تتلقّى اسم الطالب وعدد المواضيع التي تعلّمها الطالب، وتبتدىء صفات الفئة. المصفوفة subArray مُبتَدَأة لمصفوفة فارغة بِكِبَر num.
public double Average ()	عمليّة تُعيد معدّل جميع العلامات في شهادة الطالب.
public bool IsExcellent ()	عمليّة تُعيد true إذا اعتُبر الطالب طالبًا متميّزًا، خلاف ذلك تُعيد العمليّة false . الطالب المتميّز: الطالب الذي معدّل علاماته هو 85 أو أعلى، وجميع علاماته هي أعلى من 54، وإحدى علاماته على الأقلّ، هي 100.

(انتبه: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

أ. طبِّق العمليّة البنائيّة للفئة **ReportCard**.

ب. طبِّق العمليّة IsExcellent في الفئة **ReportCard**.

يمكن استعمال العمليّة Average للفئة **ReportCard** بدون تطبيقها.

افترِضْ أنّ حدود المصفوفة subArray تختلف عن null.

جـ. اكتب عمليّة خارجيّة اسمها PrintExcellent تتلقّى مصفوفة كائنات — array من نمط **ReportCard**، تمثّل طلّابًا في صفّ معيّن. تطبع العمليّة أسماء الطلّاب المتميّزين.

يجب استعمال العمليّة IsExcellent للفئة **ReportCard**.

افترِضْ أنّ حدود المصفوفة array تختلف عن null.

/ يتبع في صفحة 8 /

3. انتبه! لهذا السؤال صيغتان: بلغة Java في الصفحتين 8-9 وبلغة #C في الصفحتين 10-11.

للّذين يحلّون بلغة Java

"نصّ خاصّ" هو نصّ تظهر فيه الرموز المتطابقة بتسلسل أو نصّ بدون رموز متطابقة (بما في ذلك النصّ الفارغ).

أمثلة لِـ "نصّ خاصّ": "**1zz" ، "*&$" ، "a" ، "aaaa" ، "ccabbb".

أمثلة لنصوص ليست "نصّ خاصّ": "**33%3" ، "aabcb" ، "aba".

معطاة الفئة **MyString**.

```
public class MyString {
  private String str;
}
```

أمامك واجهة تطبيق الفئة **MyString**.

يجب استعمال عمليّات واجهة التطبيق بدون حاجة لتطبيقها.

اسم العمليّة	عنوان العمليّة
عمليّة تبني نصًّا فارغًا.	public MyString ()
عمليّة تُعيد عدد مواضع ظهور الرمز ch في النصّ str.	public int countChar (char ch)
عمليّة تتلقّى رمزًا ch وتمحو من النصّ str جميع مواضع ظهوره.	public void removeChar (char ch)
عمليّة تُضيف الرمز ch إلى نهاية النصّ str.	public void appendChar (char ch)
عمليّة تُعيد الرمز الأوّل في النصّ str.	public char firstChar ()
عمليّة تُعيد true إذا كان النصّ str فارغًا، خلاف ذلك تُعيد العمليّة false.	public boolean isEmpty ()

(انتبه: تكملة السؤال في الصفحة التالية.)

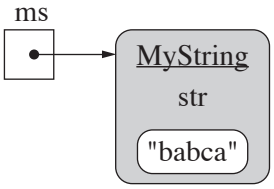
/ يتبع في صفحة 9 /

أمامك عنوان العمليّة الخارجيّة special:

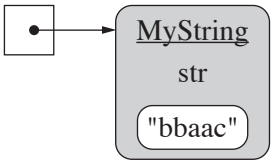
public static MyString special (MyString ms)

العمليّة special تُعيد كائنًا جديدًا من نمط **MyString** صفته، str ، هي "نصّ خاصّ" مركَّب من جميع الرموز التي في الصفة str للكائن ms .
طبِّق العمليّة special بواسطة عمليّات واجهة تطبيق الفئة **MyString** فقط. لا تُضِف عمليّات، ولا تُضِف أيضًا علميّتَي get وَ set، إلى الفئة **MyString**.

مثال:
بالنسبة للكائن ms الذي أمامك، الذي فيه نصّ عاديّ:



العمليّة special تُعيد كائنًا جديدًا، فيه "نصّ خاصّ".



ملاحظات:
- لا حاجة للحفاظ على النصّ str في الكائن ms.
- لا توجد أهمّيّة لترتيب الرموز في الكائن الجديد، طالما حُفِظ تعريف "نصّ خاصّ".

/ يتبع في صفحة 10 /

للّذين يحلّون بلغة C#

"نصّ خاصّ" هو نصّ تظهر فيه الرموز المتطابقة بتسلسل أو نصّ بدون رموز متطابقة (بما في ذلك النصّ الفارغ).

أمثلة لِـ "نصّ خاصّ": "**1zz" ، "*&$" ، "a" ، "aaaa" ، "ccabbb".

أمثلة لنصوص ليست "نصّ خاصّ": "**33%3" ، "aabcb" ، "aba".

معطاة الفئة **MyString**.

```
public class MyString {
   private string str;
}
```

أمامك واجهة تطبيق الفئة **MyString**.

يجب استعمال عمليّات واجهة التطبيق بدون حاجة لتطبيقها.

اسم العمليّة	عنوان العمليّة
عمليّة تبني نصًّا فارغًا.	public MyString ()
عمليّة تُعيد عدد مواضع ظهور الرمز ch في النصّ str.	public int CountChar (char ch)
عمليّة تتلقّى رمزًا ch وتمحو من النصّ str جميع مواضع ظهوره.	public void RemoveChar (char ch)
عمليّة تُضيف الرمز ch إلى نهاية النصّ str.	public void AppendChar (char ch)
عمليّة تُعيد الرمز الأوّل في النصّ str.	public char FirstChar ()
عمليّة تُعيد true إذا كان النصّ str فارغًا، خلاف ذلك تُعيد العمليّة false.	public bool IsEmpty ()

(انتبه: تكملة السؤال في الصفحة التالية.)

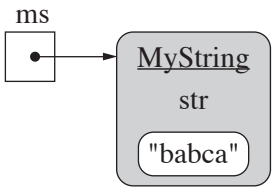
/ يتبع في صفحة 11 /

أمامك عنوان العمليّة الخارجيّة Special:

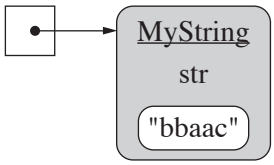
public static MyString Special (MyString ms)

العمليّة Special تُعيد كائنًا جديدًا من نمط **MyString** صفته، str ، هي "نصّ خاصّ" مركَّب من جميع الرموز التي في الصفة str للكائن ms .
طبِّق العمليّة Special بواسطة عمليّات واجهة تطبيق الفئة **MyString** فقط. لا تُضِف عمليّات، ولا تُضِف أيضًا علميّتَي Get وَ Set، إلى الفئة **MyString**.

مثال:
بالنسبة للكائن ms الذي أمامك، الذي فيه نصّ عاديّ:



العمليّة Special تُعيد كائنًا جديدًا، فيه "نصّ خاصّ" .



ملاحظات:
- لا حاجة للحفاظ على النصّ str في الكائن ms.
- لا توجد أهمّيّة لترتيب الرموز في الكائن الجديد، طالما حُفِظ تعريف "نصّ خاصّ" .

/ يتبع في صفحة 12 /

الفصل الثاني

انتبه: في كلّ سؤال يُطلب فيه تطبيق، بإمكانك استعمال عمليّات الفئات: دور، وراصّة، وشجرة بينارية وحلقة، بدون تطبيقها. إذا استعملتَ عمليّات إضافيّة، عليك تطبيقها.

4. بإمكانك في هذا السؤال استعمال العمليّة الخارجيّة التي أمامك بدون تطبيقها.

وصف العمليّة	عنوان العمليّة
تتلقّى العمليّة عددًا — num وتوجيهًا إلى بداية سلسلة حلقات — lst. تمحو العمليّة الحلقات التي فيها القيمة num وتُعيد توجيهًا إلى بداية سلسلة الحلقات.	بلغة **Java** public static Node<Integer> delete (int num, Node<Integer> lst) بلغة **C#** public static Node<int> Delete (int num, Node<int> lst)

معطاة الفئة **BiList** — سلسلة ثنائيّة، لها صفتان:

- lst1 — توجيه إلى بداية سلسلة حلقات من نمط صحيح
- lst2 — توجيه إلى بداية سلسلة حلقات من نمط صحيح

أمامك واجهة تطبيق جزئيّة للفئة **BiList** باللغتين **Java** وَ **C#**.
يجب استعمال عمليّات واجهة التطبيق بدون حاجة لتطبيقها.

وصف العمليّة	عنوان العمليّة
عمليّة تبني الكائن مع توجيهات إلى سلسلتين فارغتين.	public BiList ()
عمليّة تُضيف حلقة فيها القيمة num إلى نهاية السلسلة lst1 أو إلى نهاية السلسلة lst2 وفقًا لـ codeList: عندما codeList = 1، يُدخَل num إلى lst1، وعندما codeList = 2، يُدخَل num إلى lst2. افترِضْ أنّ قيمة البارامتر codeList سليمة.	بلغة **Java** public void addNum (int num, int codeList) بلغة **C#** public void AddNum (int num, int codeList)

(انتبه: تكملة السؤال في الصفحة التالية.)

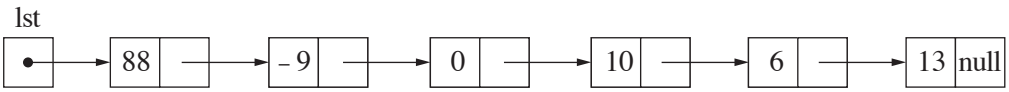
/ يتبع في صفحة 13 /

اكتب عمليّة خارجيّة اسمها generateBilist بلغة Java أو GenerateBilist بلغة C# تتلقّى سلسلة حلقات — lst لأعداد صحيحة. عدد الحلقات في lst زوجيّ والأعداد في حلقاتها تختلف عن بعضها البعض.
تُعيد العمليّة كائنًا من نمط **BiList** تتحقّق فيه الشروط التالية:

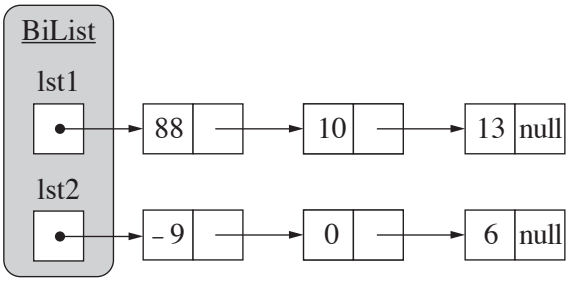
- كلّ واحد من الأعداد في السلسلة lst يظهر في إحدى السلسلتين lst1 وَ lst2.
- جميع الأعداد في السلسلة lst1 تكون أكبر من جميع الأعداد في السلسلة lst2.
- عدد الحلقات في السلسلتين lst1 وَ lst2 يكون متطابقًا.

انتبه: لا تُضِف عمليّات، ولا تُضِف أيضًا عمليّتَي get وَ set بلغة Java أو Get وَ Set بلغة C#، إلى الفئة **BiList**.

مثال:
بالنسبة للسلسلة lst التي أمامك:



تُعيد العمليّة هذا الكائن:



ملاحظات:

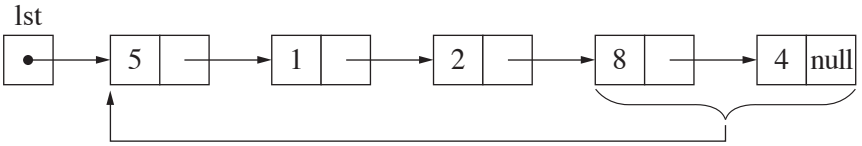
- لا حاجة للحفاظ على السلسلة lst.
- لا توجد أهمّيّة لترتيب الحدود في السلسلة lst1 وفي السلسلة lst2.

/ يتبع في صفحة 14 /

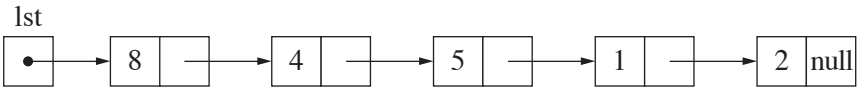
5. في سلسلة حلقات، "نقل دائريّ لِـ n حلقات" هو نقل n الحلقات الأخيرة إلى بداية السلسلة (بدون تغيير ترتيب ظهورها).

في المثال الذي أمامك بالنسبة لِـ $n = 2$: ننقل الحلقتين الأخيرتين إلى بداية السلسلة.

سلسلة الحلقات قبل النقل



سلسلة الحلقات بعد النقل



أ. اكتب عمليّة خارجيّة اسمها move بلغة Java أو Move بلغة C# تتلقّى سلسلة حلقات – lst من نمط صحيح وعددًا n من نمط صحيح. تُعيد العمليّة سلسلة الحلقات بعد "نقل دائريّ لِـ n حلقات".
افترِضْ أنّ: $n \geq 0$ وأنّ عدد الحلقات في السلسلة هو أكبر من n.

ب. ما هي تعقيدات زمن تشغيل العمليّة التي كتبتَها في البند "أ"؟ علّل.

/ يتبع في صفحة 15 /

6. انتبه ! لهذا السؤال صيغتان : بلغة Java في صفحة 15 وبلغة C# في صفحة 16 .

بلغة Java :

أ. أمامك العمليّة sod1 .

```
public static boolean sod1 (int[] arr, int x, int i)
{
    if ( i == −1) return false;
    if (arr[i] == x) return true;
    return sod1(arr, x, i −1);
}
```

(1) اكتب القيمة التي تُعاد بالنسبة لاستدعاء العمليّة sod1(a, 8, a.length − 1) والمصفوفة a التي أمامك .
عليك تبيين متابعة .

	0	1	2	3	4
a	5	4	15	12	2

(2) بالنسبة لمصفوفة ما a وعدد x ، ما هو هدف العمليّة sod1(a, x, a.length − 1)؟

(3) ما هي تعقيدات زمن تشغيل العمليّة sod1؟ علّل .

ب. أمامك العمليّة sod2 .

```
public static  boolean sod2 (int[] arr, int x, int i)
{
    if ( i == 0) return false;
    if (sod1(arr, x – arr[i], i −1)) return true;
    return sod2(arr, x, i −1);
}
```

(1) اكتب القيمة التي تُعاد بالنسبة لاستدعاء العمليّة sod2(a, 16, a.length − 1) والمصفوفة a التي أمامك .
عليك تبيين متابعة .

	0	1	2	3	4
a	5	4	15	12	2

في هذا البند، لا حاجة لتنفيذ متابعة للعمليّة sod1 .

(2) بالنسبة لمصفوفة ما a وعدد x ، ما هو هدف العمليّة sod2(a, x, a.length − 1)؟

(3) ما هي تعقيدات زمن تشغيل العمليّة sod2؟ علّل .

/ يتبع في صفحة 16 /

بلغة C#

أ. أمامك العمليّة Sod1.

```
public static bool Sod1 (int[] arr, int x, int i)
{
    if ( i == −1) return false;
    if (arr[i] == x) return true;
    return Sod1(arr, x, i − 1);
}
```

(1) اكتب القيمة التي تُعاد بالنسبة لاستدعاء العمليّة Sod1(a, 8, a.Length − 1) والمصفوفة a التي أمامك.
عليك تبيين متابعة.

	0	1	2	3	4
a	5	4	15	12	2

(2) بالنسبة لمصفوفة ما a وعدد x ، ما هو هدف العمليّة Sod1(a, x, a.Length − 1)؟

(3) ما هي تعقيدات زمن تشغيل العمليّة Sod1؟ علّل.

ب. أمامك العمليّة Sod2.

```
public static  bool Sod2 (int[] arr, int x, int i)
{
    if ( i == 0) return false;
    if (Sod1(arr, x – arr[i], i − 1)) return true;
    return Sod2(arr, x, i − 1);
}
```

(1) اكتب القيمة التي تُعاد بالنسبة لاستدعاء العمليّة Sod2(a, 16, a.Length − 1) والمصفوفة a التي أمامك.
عليك تبيين متابعة.

	0	1	2	3	4
a	5	4	15	12	2

في هذا البند، لا حاجة لتنفيذ متابعة للعمليّة Sod1 .

(2) بالنسبة لمصفوفة ما a وعدد x ، ما هو هدف العمليّة Sod2(a, x, a.Length − 1)؟

(3) ما هي تعقيدات زمن تشغيل العمليّة Sod2؟ علّل.

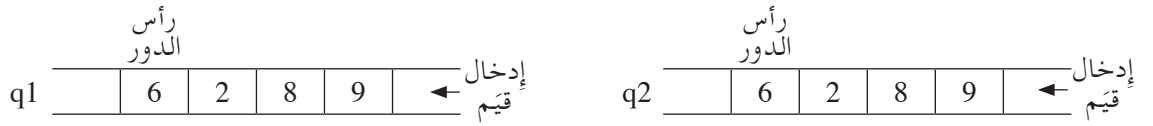
/ يتبع في صفحة 17 /

7. بإمكانك في هذا السؤال استعمال العمليّة الخارجيّة التي أمامك بدون تطبيقها.

وصف العمليّة	عنوان العمليّة
تُعيد العمليّة عدد الحدود في الدور q بدون تغيير الدور.	بلغة **Java** — public static int size (Queue<Integer> q) بلغة **C#** — public static int Size (Queue<int> q)

أ. يكون الدوران، q1 وَ q2 ، "دورين متطابقين" إذا كان عدد الحدود في الدورين متطابقًا، وتظهر في الدورين بالضبط نفس القيَم وبنفس الترتيب.

مثال لدورين متطابقين:

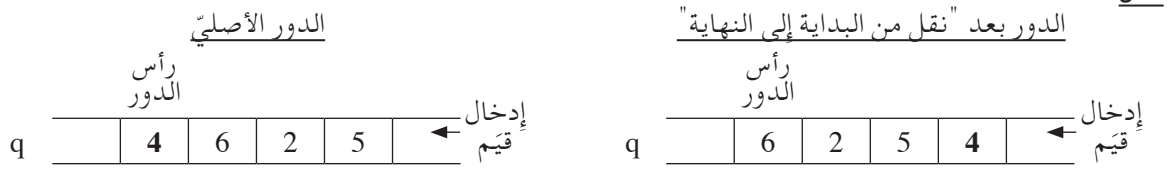


اكتب عمليّة خارجيّة اسمها isIdentical بلغة Java أو IsIdentical بلغة C# تتلقّى دورين من نمط صحيح q1 وَ q2 ، وتُعيد true إذا كان الدوران متطابقين، خلاف ذلك تُعيد العمليّة false.

ملاحظة: عند انتهاء العمليّة، يجب الحفاظ على مبنى الدورين الأصليّ الذي تمّ تلقّيه.

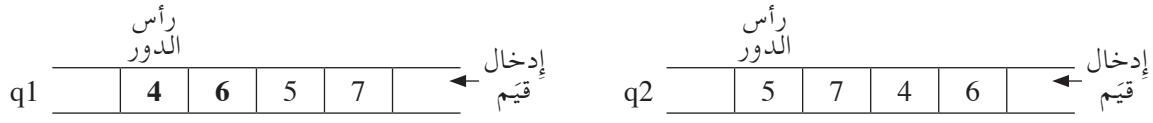
ب. "نقل من البداية إلى النهاية" هو نقل عدد من رأس الدور إلى نهايته.

مثال:

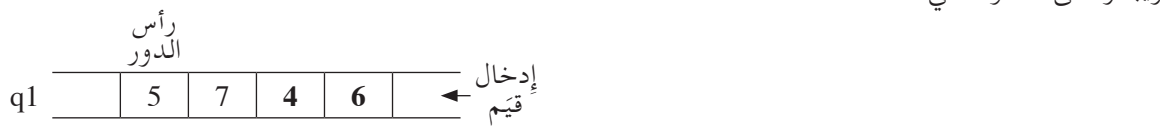


اكتب عمليّة خارجيّة اسمها isSimilar بلغة Java أو IsSimilar بلغة C# تتلقّى دورين من نمط صحيح، q1 وَ q2 . تُعيد العمليّة true إذا كان الدوران q1 وَ q2 متطابقين – سواءً كانا متطابقين كما هما، أو أنّهما سيكونان متطابقين بعد أن يُنفَّذ في q1 "نقل من البداية إلى النهاية"، مرّة واحدة أو أكثر. خلاف ذلك تُعيد العمليّة false.

مثال: تُعيد العمليّة true بالنسبة للدورين q1 وَ q2 اللذين أمامك:



وذلك لأنّه بعد أن ننفّذ مرّتين في الدور q1 "نقل من البداية إلى النهاية"، فإنّه سيكون مطابقًا للدور q2، ويبدو على النحو التالي:



ملاحظات: – يجب استعمال العمليّة التي كتبتَها في البند "أ".
– لا حاجة للحفاظ على مبنى الدورين الأصليّ الذي تمّ تلقّيه.

/ يتبع في صفحة 18 /

الفصل الثالث

في هذا الفصل أسئلة في أربعة مسارات:
أنظمة حاسوب وأسمبلي، في الصفحتين 18-19.
مقدّمة لبحث الأداءات، في الصفحات 20-22.
موديلات حسابيّة، في الصفحة 23 .
برمجة موجّهة كائنات بلغة Java ، في الصفحات 24-27؛ برمجة موجّهة كائنات بلغة #C، في الصفحات 28-31.

إذا اخترتَ الإجابة عن أسئلة من هذا الفصل، اختر أسئلة من مسار واحد فقط.

أنظمة حاسوب وأسمبلي

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

أ. أمامك قطعة برنامج بلغة أسمبلي.

```
    MOV   AL , 6
    MOV   BX , 5CH
    MOV   DL , [BX]
    DEC   BX
A1: MOV   CL , [BX]
    INC   CL
    MOV   [BX+1] , CL
    DEC   BX
    DEC   AL
    CMP   AL , 0
    JA    A1
    INC   BX
    INC   DL
    MOV   [BX] , DL
```

معطاة خارطة خلايا الذاكرة من 56H وحتّى 5CH قبل تنفيذ قطعة البرنامج:

address	56H	57H	58H	59H	5AH	5BH	5CH
value	59H	E2H	C5H	71H	ABH	3FH	8DH

(1) تتبّع بواسطة جدول متابعة قطعة البرنامج. يجب أن يشمل جدول المتابعة عمودًا لكلّ واحد من الخوازن BX ، AL ، CL ، DL. بالإضافة إلى ذلك، يجب كتابة خارطة ذاكرة ملائمة.

(2) اشرح ما الذي تنفّذه قطعة البرنامج.

ب. في مقطع المعطيات عُرِّف المتغيّر REZ بِكبَر كلمة:

```
REZ DW ?
```

معطى أنّه في الراصّة (في SP وفي SP+2) مخزون عددان موجَّهان (signed) بكِبَر كلمة.
اكتب قطعة برنامج تخزن في المتغيّر REZ العدد الذي قيمته المطلقة هي الأكبر من بينهما.
ملاحظات: – افترِضْ أنّ العددين يختلفان عن 8000H .
– افترِضْ أنّ القيمة المطلقة للعددين ليست متطابقة.

/ يتبع في صفحة 19 /

9. في مقطع المعطيات عُرِّفت المعطيات التي أمامك:

```
ARR DB 10 DUP (?)
VAL DB ?
IND DB ?
```

اكتب قطعة برنامج، تُزاح فيها الحدود التي في المصفوفة – ARR ابتداءً من المؤشّر IND وحتّى نهاية المصفوفة، إلى اليمين بحيث يخرج الحدّ الأيمن الأقصى من المصفوفة، وتُدخَل القيمة VAL في المؤشّر IND .
افترِضْ أنّ المصفوفة مليئة بالأعداد، وأنّ VAL مُبتدَأ وأنّ $0 \le IND \le 9$.

مثال: بالنسبة لِـ: **VAL = 15** وَ **IND = 5** والمصفوفة – ARR التالية:

index	0	1	2	3	4	**5**	6	7	8	9
value	−3	11	0	1	8	**−9**	**5**	**6**	**33**	**73**

تنتج بعد تنفيذ قطعة البرنامج المصفوفة – ARR التالية:

index	0	1	2	3	4	**5**	6	7	8	9
value	−3	11	0	1	8	**15**	**−9**	**5**	**6**	**33**

/ يتبع في صفحة 20 /

مقدّمة لبحث الأداءات

10. أ. **(1)** في الجدول الذي أمامك معطاة مسألة نقل.

العرض	الغايات				المصادر
	4	3	2	1	
200	[3]	[9]	[7]	[10]	1
300	[5]	[4]	[12]	[15]	2
500	[7]	[6]	[13]	[17]	3
	80	250	150	100	**الطلب**

انسخ الجدول إلى دفترك، واقترِحْ حلّاً أساسيًّا ممكنًا حسب طريقة الزاوية الشماليّة – الغربيّة.

(2) في الجدول الذي أمامك معطى حلّ أساسيّ ممكن لمسألة نقل، ومعطاة قيمة u_1 .

u_i	العرض	الغايات				المصادر
		4	3	2	1	
0	200	[3]	[9]	[7] (100)	[10] (100)	1
	300	[5] (250)	[4]	[12] (50)	[15]	2
	200	[7] (180)	[6] (20)	[13]	[17]	3
		180	270	150	100	**الطلب**
						v_j

I. انسخ الجدول إلى دفترك، وأكمل فيه القيَم v_4 ، v_3 ، v_2 ، v_1 ، u_3 ، u_2 .

II. برهن أنّ الحلّ المعطى في الجدول ليس أمثل.

III. غيِّر السعر (c_{ij}) في أحد المتغيّرات غير الأساسيّة بحيث الحلّ المعطى في الجدول لمسألة النقل يكون أمثل.

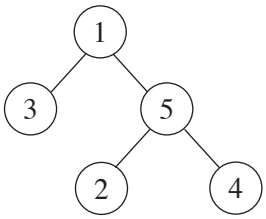
(انتبه: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 21 /

ب. الرسم البيانيّ $G = (V, E)$ غير الموجَّه والقابل للارتباط والبسيط، يُسمّى "منقسمًا" إذا تحقّق فيه الشرطان التاليان:

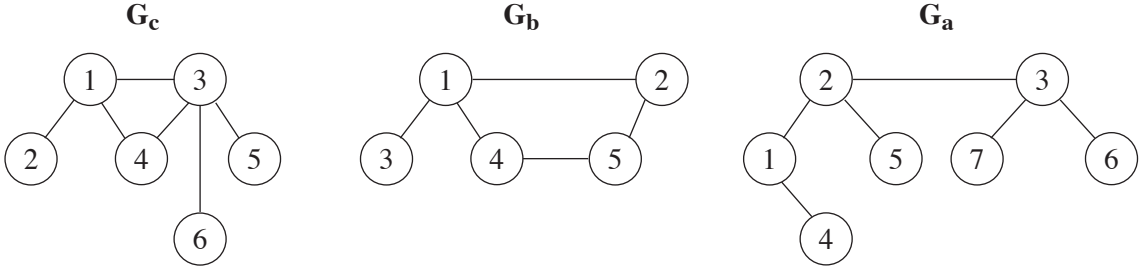
- توجد فيه عقدتان على الأقلّ.
- يمكن تقسيم العُقَد إلى مجموعتين، المجموعة "أ" والمجموعة "ب"، بطريقة لا تكون فيها عقدتان مرتبطتين بقوس في نفس المجموعة.

مثال:



هذا الرسم البيانيّ هو رسم بيانيّ "منقسم" لأنّه يمكن تقسيم العُقَد بطريقة تحوي فيها المجموعة "أ" العُقدتين $\{3, 5\}$ وتحوي فيها المجموعة "ب" العُقَد $\{1, 2, 4\}$.

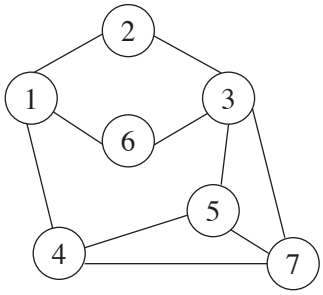
(1) أمامك ثلاثة رسوم بيانيّة: G_a ، G_b ، G_c .



اذكر أيًّا من الرسوم البيانيّة هو رسم بيانيّ "منقسم" وأيًّا منها ليس رسمًا بيانيًّا "منقسمًا". بالنسبة لكلّ رسم بيانيّ "منقسم"، بيّن مجموعتَي العُقَد التي فيه.

(2) أمامك رسم بيانيّ <u>ليس</u> "منقسمًا". ما هو عدد الأقواس الأدنى التي يجب إزالتها كي يُعتبَر الرسم البيانيّ "منقسمًا"؟

اكتب أيّة أقواس يجب إزالتها، واعرض المجموعتين.



/ يتبع في صفحة 22 /

11. في هذا السؤال بندان، "أ – ب"، لا علاقة بينهما. أجب عند البندين.

أ. أمامك مصفوفة مجاوَرات تصف رسمًا بيانيًّا موجَّهًا $G=(V,E)$.

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

(1) ارسم الرسم البيانيّ G بصورة بيانيّة.

(2) جد مركِّبات الربط القويّ في الرسم البيانيّ.

(3) ما هو عدد الأقواس الأدنى التي يجب إضافتها إلى الرسم البيانيّ G كي يتكوّن مركّب ربط قويّ واحد فقط؟ اذكر الأقواس التي يجب إضافتها.

ب. أمامك مصفوفة مجاوَرات تصف وزن مسار بين عُقَد في رسم بيانيّ موجَّه وموزون (משוקלל) $G=(V,E)$.

	a	**b**	**c**	**d**	**e**
a	0	1	3	∞	∞
b	2	0	1	3	12
c	3	∞	0	5	∞
d	4	∞	∞	0	7
e	1	2	∞	8	0

معطاة نقطتا بداية ممكنتان: a أو c. عليك أن تصل إلى العقدة e من إحدى هاتين النقطتين.

(1) ما هي نقطة البداية التي يكون منها وزن المسار إلى العقدة e هو الأدنى؟

عليك تفعيل ألغوريثم دكسترا وتبيين متابعة في كلّ تكرار.

تشمل المتابعة في كلّ تكرار مجموعة العُقَد الثابتة (P) ومجموعة العُقَد المؤقّتة (T).

(2) إذا غيّرنا وزن القوس من العقدة a إلى العقدة c إلى الوزن k ، $k \geq 0$ ، هل يتغيّر الحلّ؟ فسّر.

/ يتبع في صفحة 23 /

موديلات حسابيّة

12. L هي لغة جميع الكلمات فوق الأبجديّة {a,b,c} التي تحقّق جميع المتطلّبات التي أمامك:

- الحرف قبل الأخير في الكلمة هو a .
- تحوي الكلمة على الأقلّ مرّتين التسلسل bc .
- عدد مرّات ظهور الحرف b في الكلمة هو زوجيّ.
- الكلمة لا تحوي التسلسل bb .

أ. **(1)** أعطِ مثالاً واحدًا لكلمة تتبع للّغة L .

(2) أعطِ مثالاً واحدًا لكلمة لا تتبع للّغة L .

ب. برهن أنّ اللّغة L هي نظاميّة.

ملاحظة: بإمكانك الاستعانة بصفات انغلاق.

13. معطاة اللّغتان L_1 وَ L_2 :

$L_1 = \{ a^n b^m c^k \mid n, k \geq 0, m = 2k \}$

$L_2 = \{ a^n b^m c^k \mid$ {k , m , n} أكبر من 0 ويجب أن تكون إمّا جميعها زوجيّة وإمّا جميعها فرديّة $\}$

أ. هل اللّغة L_1 نظاميّة؟ إذا كانت نظاميّة، ابنِ أوتوماتًا نهائيًّا محدودًا كاملاً يتلقّى اللّغة، وإذا لم تكن نظاميّة، ابنِ أوتومات راصّة يتلقّى اللّغة.

ب. هل اللّغة L_2 نظاميّة؟ إذا كانت نظاميّة، ابنِ أوتوماتًا نهائيًّا محدودًا كاملاً يتلقّى اللّغة، وإذا لم تكن نظاميّة، ابنِ أوتومات راصّة يتلقّى اللّغة.

/ يتبع في صفحة 24 /

برمجة موجّهة كائنات بلغة Java

السؤالان 14-15 معدّان للّذين يكتبون بلغة Java.

14. بهدف إدارة أحداثها، بَنَت شركة "الغد" الفئة **Date** – تاريخ، التي لها ثلاث صفات من نمط صحيح:
يوم – day ، شهر – month ، سنة – year .
بالإضافة إلى ذلك، بهدف إدارة الأحداث بُنيت الفئة **Event** – حدث.
للشركة ثلاثة أنواع أحداث: لقاءات ومكالمات هاتفيّة ومهمّات. من أجل حفظ المعلومات التي في كلّ نوع حدث، بنت الشركة الفئات التالية: لقاء – **Meeting** ، مكالمة هاتفيّة – **PhoneCall** ، مهمّة – **Task** .
صفات الفئات هي وفقًا لمتطلّبات المعلومات التي أمامك:

المعلومات التي تُحفَظ عن اللقاء – **Meeting**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
arrNames – أسماء المشاركين في اللقاء، في مصفوفة من نمط نصّ (كِبَر المصفوفة هو حسب عدد المشاركين)
duration – مدّة اللقاء بالدقائق، من نمط صحيح
location – مكان اللقاء، من نمط نصّ

المعلومات التي تُحفَظ عن المكالمة الهاتفيّة – **PhoneCall**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
phoneNumber – رقم الهاتف المتّصَل به، من نمط نصّ
name – اسم الشخص المتّصَل به، من نمط نصّ

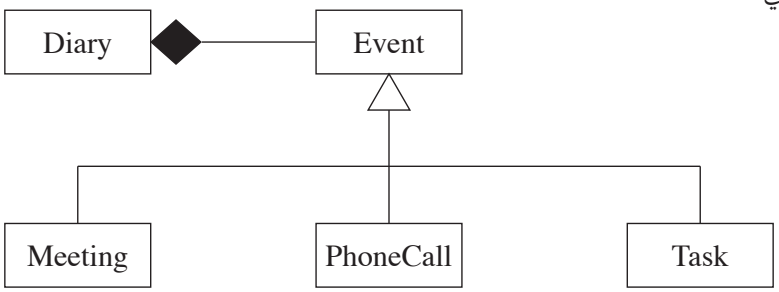
المعلومات التي تُحفَظ عن المهمّة – **Task**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
title – عنوان المهمّة، من نمط نصّ

المعلومات التي تُحفَظ عن الحدث – **Event**:
نفس متطلّبات المعلومات المفصَّلة في أنواع الأحداث أعلاه والتي يجب أن تُحفَظ في الفئة Event حسب مبادئ البرمجة الموجّهة كائنات.

بالإضافة إلى الفئات المذكورة، بُنيت الفئة **Diary** – يوميّات، التي تحفظ حتّى 1,000 حدث – **Event** ، في المصفوفة – arr بكِبَر 1,000 من نمط **Event** .

/ يتبع في صفحة 25 /

أمامك مخطّط جزئيّ للفئات:



انتبه: أُشير إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ◆——

أجب عن البنود التي أمامك حسب مبادئ البرمجة الموجّهة كائنات.

أ. انسخ المخطّط إلى دفترك، وأَضِف الفئة **Date** إلى المخطّط. اكتب عنوان الفئة حدث – **Event** وصفاتها.
افترِض أنّه في باقي الفئات مكتوبة الصفات الملائمة، وأنّه في جميع الفئات توجد عمليّات بنائيّة وعمليّتا get وَ set.

ب. اكتب في الفئة **Diary** عمليّة اسمها allCalls تتلقّى كائنًا – date من نمط **Date**، وتُعيد مصفوفة من نمط **PhoneCall** بكِبَر 100، تحوي جميع المكالمات الهاتفيّة في التاريخ – date.

انتبه:

– افترِضْ أنّ العمليّة التي أمامك موجودة في الفئة **Date**:
public boolean same (Date other)

تُعيد العمليّة true إذا كان التاريخ الحاليّ مطابقًا لِـ other، خلاف ذلك تُعيد العمليّة false.
بإمكانك استعمال العلميّة بدون تطبيقها.

– افترِضْ أنّه توجد بالضبط 100 مكالمة هاتفيّة في التاريخ – date.

– افترِضْ أنّه أثناء استدعاء هذه العمليّة، مصفوفة الأحداث – arr في الفئة **Diary** مليئة بالكائنات (بدون null).

جـ. أمامك عنوان العمليّة الداخليّة – match في الفئة **Event**:
public boolean match (String name)

يُعيد استدعاء العمليّة true إذا كان الحدث (**Event**) لقاءً (**Meeting**) يشارك فيه شخص اسمه مطابق لِـ name أو إذا كان الحدث مكالمةً هاتفيّةً (**PhoneCall**) مع شخص اسمه مطابق لِـ name، خلاف ذلك تُعيد العمليّة false.
طبِّق العمليّة، وأَضِف عمليّات إلى الفئات الملائمة من أجل تنفيذ المطلوب.
بالنسبة لكلّ عمليّة، اذكر في أيّة فئة تُضاف.

ملاحظة: **في هذا البند، لا** تَستعمِل العمليّة instanceof وعمليّات الفئة Object.

/ يتبع في صفحة 26 /

15. أمامك الفئات A ، B ، Tester :

```
public class A {
    public A() { System.out.println("ctor A"); }
    public void foo(int x) {
        System.out.println("A foo int " + x); }
    public void foo(double y) {
        System.out.println("A foo double " + y); }
    public void bar(int x) {
        System.out.println("A bar " + x);
        foo(x);
    }
}

public class B extends A {
    public B() { System.out.println("ctor B"); }
    public void foo(int x) {
        System.out.println("B foo int " + x); }
    public void bar() {
        System.out.println("B bar");
        foo(2);
    }
    public void another(int x) {
        System.out.println("B another " + x);
        super.foo(x);
        foo(2.0 * x);
    }
}

public class Tester {
    public static void main (String [] args)
    {
    }
}
```

(انتبه: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 27 /

أمامك 15 قطعة كود. اختر عشرًا منها.

عوِّض كلّ واحدة من قِطَع الكود التي اخترتَها في العمليّة main في الفئة Tester ، اكتب في دفترك رقم القطعة، واذكر إذا كان الكود سليمًا أم غير سليم. إذا كان الكود سليمًا – اكتب المخرَج، وإذا لم يكن سليمًا – اشرح الخطأ.

قِطَع الكود لا يرتبط (يتعلّق) بعضها ببعض.

1. A a = new A(); a.foo(2);	**2.** A a = new A(); ((B)a).foo(3);	**3.** A a = new A(); B b = a; b.foo(2);
4. A x = new B(); x.foo(2);	**5.** A x = new B(); x.bar();	**6.** A a = new A(); a.bar(3);
7. A a = new A(); a.bar();	**8.** B b = new B(); b.bar();	**9.** B b = new B(); b.bar(3);

10.
B b = new B();
b.foo(2);
b.foo(2.0);
	11.
	A a = new A();
	a.foo(2);
	a.foo(2.0);
		12.
		B b = new A();
		b.another(2);

13. A a = new A(); a.another(2);	**14.** B x = new B(); x.another(2);	**15.** A x = new B(); x.another(2);

/ يتبع في صفحة 28 /

برمجة موجّهة كائنات بلغة #C

السؤالان 16-17 معدّان للّذين يكتبون بلغة C#.

16. بهدف إدارة أحداثها، بَنَت شركة "الغد" الفئة **Date** – تاريخ، التي لها ثلاث صفات من نمط صحيح:
يوم – day ، شهر – month ، سنة – year .
بالإضافة إلى ذلك، بهدف إدارة الأحداث بُنيت الفئة **Event** – حدث.
للشركة ثلاثة أنواع أحداث: لقاءات ومكالمات هاتفيّة ومهمّات. من أجل حفظ المعلومات التي في كلّ نوع حدث، بنت الشركة الفئات التالية: لقاء – **Meeting** ، مكالمة هاتفيّة – **PhoneCall** ، مهمّة – **Task** .
صفات الفئات هي وفقًا لمتطلّبات المعلومات التي أمامك:

المعلومات التي تُحفَظ عن اللقاء – **Meeting**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
arrNames – أسماء المشاركين في اللقاء، في مصفوفة من نمط نصّ (كِبَر المصفوفة هو حسب عدد المشاركين)
duration – مدّة اللقاء بالدقائق، من نمط صحيح
location – مكان اللقاء، من نمط نصّ

المعلومات التي تُحفَظ عن المكالمة الهاتفيّة – **PhoneCall**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
phoneNumber – رقم الهاتف المتّصَل به، من نمط نصّ
name – اسم الشخص المتّصَل به، من نمط نصّ

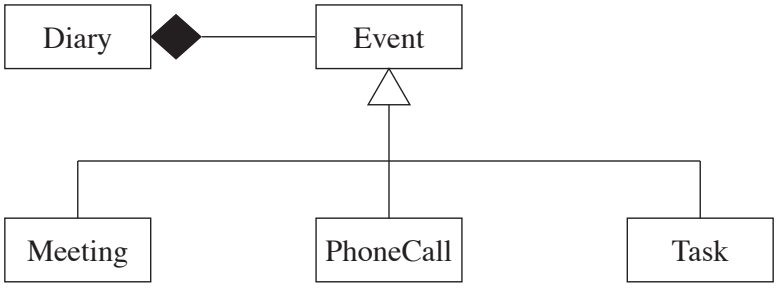
المعلومات التي تُحفَظ عن المهمّة – **Task**:
date – من نمط **Date**
hour – ساعة البداية، من نمط صحيح
title – عنوان المهمّة، من نمط نصّ

المعلومات التي تُحفَظ عن الحدث – **Event**:
نفس متطلّبات المعلومات المفضَّلة في أنواع الأحداث أعلاه والتي يجب أن تُحفَظ في الفئة Event حسب مبادئ البرمجة الموجّهة كائنات.

بالإضافة إلى الفئات المذكورة، بُنيت الفئة **Diary** – يوميّات، التي تحفظ حتّى 1,000 حدث – **Event** ، في المصفوفة – arr بكِبَر 1,000 من نمط **Event** .

/ يتبع في صفحة 29 /

أمامك مخطّط جزئيّ للفئات:



انتبه: أُشير إلى توريث بواسطة السهم ◁—— وإلى احتواء بواسطة الإشارة ◆——

أجب عن البنود التي أمامك حسب مبادئ البرمجة الموجّهة كائنات.

أ. انسخ المخطّط إلى دفترك، وأَضِف الفئة **Date** إلى المخطّط. اكتب عنوان الفئة حدث – **Event** وصفاتها.
افترِض أنّه في باقي الفئات مكتوبة الصفات الملائمة، وأنّه في جميع الفئات توجد عمليّات بنائيّة وعمليّتا Get وَ Set.

ب. اكتب في الفئة **Diary** عمليّة اسمها AllCalls تتلقّى كائنًا – date من نمط **Date**، وتُعيد مصفوفة من نمط **PhoneCall** بكِبَر 100، تحوي جميع المكالمات الهاتفيّة في التاريخ – date.

انتبه:

– افترِضْ أنّ العمليّة التي أمامك موجودة في الفئة **Date**:
public bool Same (Date other)
تُعيد العمليّة true إذا كان التاريخ الحاليّ مطابقًا لِـ other، خلاف ذلك تُعيد العمليّة false.
بإمكانك استعمال العلميّة بدون تطبيقها.

– افترِضْ أنّه توجد بالضبط 100 مكالمة هاتفيّة في التاريخ – date.

– افترِضْ أنّه أثناء استدعاء هذه العمليّة، مصفوفة الأحداث – arr في الفئة **Diary** مليئة بالكائنات (بدون null).

جـ. أمامك عنوان العمليّة الداخليّة – Match في الفئة **Event**:
public virtual bool Match (string name)
يُعيد استدعاء العمليّة true إذا كان الحدث (**Event**) لقاءً (**Meeting**) يشارك فيه شخص اسمه مطابق لِـ name أو إذا كان الحدث مكالمةً هاتفيّةً (**PhoneCall**) مع شخص اسمه مطابق لِـ name، خلاف ذلك تُعيد العمليّة false.

طبِّق العمليّة، وأَضِف عمليّات إلى الفئات الملائمة من أجل تنفيذ المطلوب.
بالنسبة لكلّ عمليّة، اذكر في أيّة فئة تُضاف.

ملاحظة: **في هذا البند، لا** تَستعمِل العمليّتين is وَ as وعمليّات الفئة Object.

/ يتبع في صفحة 30 /

17. أمامك الفئات A ، B ، Tester :

```
public class A {
    public A() { Console.WriteLine("ctor A"); }
    public virtual void Foo(int x) {
        Console.WriteLine("A Foo int " + x); }
    public void Foo(double y) {
        Console.WriteLine("A Foo double " + y); }
    public void Bar(int x) {
        Console.WriteLine("A Bar " + x);
        Foo(x);
    }
}

public class B : A {
    public B() { Console.WriteLine("ctor B"); }
    public override void Foo(int x) {
        Console.WriteLine("B Foo int " + x); }
    public void Bar() {
        Console.WriteLine("B Bar");
        Foo(2);
    }
    public void Another(int x) {
        Console.WriteLine("B Another " + x);
        base.Foo(x);
        Foo(2.0 * x);
    }
}

public class Tester {
    public static void Main (string [] args)
    {
    }
}
```

(انتبه: تكملة السؤال في الصفحة التالية.)

/يتبع في صفحة 31/

أمامك 15 قطعة كود. اختر عشرًا منها.

عوِّض كلّ واحدة من قِطَع الكود التي اخترتَها في العمليّة Main في الفئة Tester ، اكتب في دفترك رقم القطعة، واذكر إذا كان الكود سليمًا أم غير سليم. إذا كان الكود سليمًا – اكتب المخرَج، وإذا لم يكن سليمًا – اشرح الخطأ.

قِطَع الكود لا يرتبط (يتعلّق) بعضها ببعض.

1. A a = new A(); a.Foo(2);	**2.** A a = new A(); ((B)a).Foo(3);	**3.** A a = new A(); B b = a; b.Foo(2);
4. A x = new B(); x.Foo(2);	**5.** A x = new B(); x.Bar();	**6.** A a = new A(); a.Bar(3);
7. A a = new A(); a.Bar();	**8.** B b = new B(); b.Bar();	**9.** B b = new B(); b.Bar(3);

10.
B b = new B();
b.Foo(2);
b.Foo(2.0);
	11.
	A a = new A();
	a.Foo(2);
	a.Foo(2.0);
		12.
		B b = new A();
		b.Another(2);

13. A a = new A(); a.Another(2);	**14.** B x = new B(); x.Another(2);	**15.** A x = new B(); x.Another(2);

בהצלחה!
نتمنّى لك النّجاح!

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