מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תש"ף, 2020
מספר השאלון: **899381**
תרגום לערבית (2)

מדעי המחשב

הוראות לנבחן

א. משך הבחינה: שלוש שעות.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שלושה פרקים.
פרק ראשון
(10×1)+(15×1) — 25 נק'
פרק שני
(25×2) — 50 נק'
פרק שלישי
(25×1) — 25 נק'
סה"כ — 100 נק'

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

ד. הוראות מיוחדות:
1. את כל התוכניות שאתה נדרש לכתוב בשפת מחשב בפרקים הראשון והשני, כתוב בשפה אחת בלבד — Java או #C.
2. **רשום על הכריכה החיצונית** של המחברת באיזו **שפה** אתה כותב — Java או #C.
3. **רשום על הכריכה החיצונית** של המחברת את שם ה**מסלול שלמדת**. המסלול הוא אחד מארבעת המסלולים האלה: מערכות מחשב ואסמבלי, מבוא לחקר ביצועים, מודלים חישוביים, תכנות מונחה עצמים.

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

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: صيف 2020
رقم النّموذج: **899381**
ترجمة إلى العربيّة (2)

علم الحاسوب

تعليمات للممتحَن

أ. مدّة الامتحان: ثلاث ساعات.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج ثلاثة فصول.
الفصل الأوّل
(10×1)+(15×1) – 25 درجة
الفصل الثّاني
(25×2) – 50 درجة
الفصل الثّالث
(25×1) – 25 درجة
المجموع – 100 درجة

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

د. تعليمات خاصّة:
1. اكتب جميع البرامج التي يُطلب منك كتابتها بلغة حاسوب في الفصلين الأوّل والثاني، بلغة واحدة فقط – Java أو #C.
2. **سجّل على الغلاف الخارجيّ** للدفتر بأيّة **لغة** تكتب – Java أو #C.
3. **سجّل على الغلاف الخارجيّ** للدفتر، اسم **المسار الذي تعلّمتَه**. المسار هو أحد المسارات الأربعة التالية: أنظمة حاسوب وأسمبلي، مقدّمة لبحث الأداءات، موديلات حسابيّة، برمجة موجّهة كائنات.

ملاحظة: في البرامج التي تكتبها، لن تُخصم درجات إذا كتبتَ 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 : تبدأ الأسئلة في صفحة 3 .

الأسئلة

في هذا النموذج ثلاثة فصول.

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

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

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

الفصل الأوّل (25 درجة)

أجب عن السؤال 1 – إلزاميّ (10 درجات).

1. اكتب عمليّة خارجيّة above بلغة Java أو Above بلغة C#، تتلقّى مصفوفة – arr من نمط صحيح وعددًا – num من نمط صحيح. تُعيد العمليّة المؤشّر الأوّل للخليّة التي مجموع الأعداد من بداية المصفوفة وحتّى إليها (بما في ذلك هذه الخليّة) هو أكبر من العدد num. إذا لم توجد خليّة كهذه، تُعيد العمليّة 1 –.

مثال: في المصفوفة arr التي أمامك، بالنسبة لـ num يساوي 9، تُعيد العمليّة العدد 4، لأنّه هو المؤشّر الأوّل للخليّة التي مجموع الحدود من بداية المصفوفة وحتّى إليها (بما في ذلك هذه الخليّة) هو أكبر من العدد 9.

0	1	2	3	4	5
3	−2	6	2	1	3

مثال آخر: في المصفوفة arr التي أمامك، بالنسبة لـ num يساوي 11، تُعيد العمليّة العدد 1 –، لأنّه لا توجد خليّة مجموع الأعداد من بداية المصفوفة وحتّى إليها (بما في ذلك هذه الخليّة) هو أكبر من العدد 11.

0	1	2	3	4	5
1	2	−2	0	7	3

/ يتبع في صفحة 4 /

أجب عن أحد السؤالين 2-3 (15 درجة) .

2. معطاة الفئة **Weight** – وزن، التي لها صفتان :

kilo – عدد كيلوغرامات، تتلقّى قيَمًا من 0 فصاعدًا، من نمط صحيح .

gram – عدد غرامات، تتلقّى قيَمًا من 0 حتّى 999 (بما في ذلك 0 وَ 999)، من نمط صحيح .

افترِضْ أنّه تمّ لكلّ صفة في الفئة تعريف عمليّتَي get وَ set بلغة Java وعمليّتَي Get وَ Set بلغة C# .

أمامك واجهة تطبيق الفئة **Weight** ، مكتوبة بلغة Java وبلغة C# :

وصف العمليّة	عنوان العمليّة
عمليّة بنائيّة تبتدئ الصفتين kilo وَ gram إلى صفر.	public Weight ()
عمليّة بنائيّة تتلقّى قيَمًا بالكيلوغرامات والغرامات . افترِضْ أنّ عدد الكيلوغرامات في المتغيّر kilo هو أكبر أو يساوي 0 ، وعدد الغرامات في المتغيّر gram هو بين 0 وَ 999 (بما في ذلك 0 وَ 999) .	public Weight (int kilo, int gram)
عمليّة بنائيّة تتلقّى قيمة بالغرامات . عدد الغرامات في المتغيّر totalGram يمكن أن يكون أعلى من 999 . تحسب العمليّة وتُدخِل عدد الكيلوغرامات إلى kilo وعدد الغرامات إلى gram . مثال : إذا كان عدد الغرامات في المتغيّر totalGram هو 3,215 ، فإنّ الصفة kilo تتلقّى 3 والصفة gram تتلقّى 215 . افترِضْ أنّ عدد الغرامات في المتغيّر totalGram هو أكبر أو يساوي 0 .	public Weight (int totalGram)
عمليّة تتلقّى كائنًا آخر من نمط Weight وتُضيف قِيَمَهُ إلى وزن الكائن الحاليّ . افترِضْ أنّ other ليس null .	بلغة **Java** – public void add (Weight other) بلغة **C#** – public void Add (Weight other)
عمليّة تتلقّى كائنًا آخر من نمط Weight وتُعيد true إذا كان وزن الكائن الحاليّ أصغر من الوزن الآخر . افترِضْ أنّ other ليس null .	بلغة **Java** – public boolean less (Weight other) بلغة **C#** – public bool Less (Weight other)

(انتبه : تكملة السؤال في الصفحة التالية .)

/ يتبع في صفحة 5 /

أ. طبِّق العمليّتين البنائيّتين للفئة **Weight** : public Weight (int kilo, int gram)
وَ public Weight (int totalGram) (لا حاجة لتطبيق العمليّة البنائيّة بدون بارامترات) .

ب. طبِّق العمليّتين add وَ less بلغة Java ، أو العمليّتين Add وَ Less بلغة C# .

جـ. معطاة الفئة **AllWeights** التي لها صفة واحدة: arr – مصفوفة من نمط **Weight** .
أمامك عمليّة لواجهة تطبيق الفئة **AllWeights** ، مكتوبة بلغة Java وبلغة C# :

وصف العمليّة	عنوان العمليّة
عمليّة تُعيد كائنًا من نمط Weight يساوي مجموع أوزان حدود المصفوفة. ملاحظة: لا تُغيِّر حدود المصفوفة.	بلغة **Java** – public Weight sum () بلغة **C#** – public Weight Sum ()

طبِّق العمليّة sum بلغة Java أو العمليّة Sum بلغة C# .

افترِضْ أنّ الخلايا في المصفوفة ليست null .
ملاحظة: بإمكانك استعمال كلّ واحدة من عمليّات واجهة تطبيق الفئة **Weight** .

/ يتبع في صفحة 6 /

3. في فندق الشباب "أحلام سعيدة" يوجد 3 طوابق، 1-3 . في الفندق 200 غرفة بالمجمل.
في الفندق نوعان من الغرف: غرفة زوجيّة وغرفة فرديّة. سعر الليلة في الغرفة الفرديّة هو 50 شيكلاً وفي الغرفة الزوجيّة 100 شيكل.

في المنظومة المحوسَبة للفندق معرَّفة الفئة **Room** – غرفة، التي لها ثلاث صفات:

- roomNum – رقم الغرفة مكوّن من ثلاثة أرقام، من نمط صحيح.
  الرقم الأوّل (من اليسار) يمثّل رقم الطابق الذي توجد فيه الغرفة.
  مثال: غرفة رقم 231 موجودة في الطابق رقم 2 ، والغرفة 145 موجودة في الطابق رقم 1.
- roomType – نوع الغرفة: 1 هو غرفة فرديّة، وَ 2 هو غرفة زوجيّة، من نمط صحيح.
- nightsReserved – عدد الليالي في الطلبيّة منذ الآن، من نمط صحيح.
  إذا كان nightsReserved يساوي 0 فإنّ الغرفة خالية، وإذا كان أكبر من 0 – فإنّ الغرفة مشغولة.

افترِضْ أنّه تمّ لكلّ صفة في الفئة تعريف عمليّتَي get وَ set بلغة Java وعمليّتَي Get وَ Set بلغة C#.

أ. اكتب عمليّة income بلغة Java أو Income بلغة C# ، في الفئة **Room** ، تُعيد الدَّخْل الماليّ الذي يأتي من الغرفة. حساب الدَّخْل هو عدد الليالي في الطلبيّة مضروب في سعر الغرفة.

ب. في المنظومة المحوسَبة للفندق معرَّفة فئة إضافيّة، **Hostel** – فندق، لها صفة واحدة:
allRooms – مصفوفة بِكِبَر 200 من نمط **Room** . (الغرف في المصفوفة ليست مصنَّفة حسب ترتيب معيّن).
افترِضْ أنّ الخلايا في المصفوفة ليست null .

(انتبه: تكملة السؤال في الصفحة التالية.)

/ يتبع في صفحة 7 /

أمامك واجهة تطبيق جزئيّة للفئة **Hostel** ، مكتوبة بلغة Java وبلغة C# :

وصف العمليّة	عنوان العمليّة
تجد العمليّة الغرفة الأولى في المصفوفة allRooms التي تفي بالشروط التالية: خالية ومن النوع type . تُعدِّل العمليّة في الغرفة عدد الليالي المطلوب الذي يظهر في المتغيّر nights وتُعيد رقم الغرفة. إذا لم توجد في المصفوفة غرفة خالية من النوع المطلوب، تُعيد العمليّة 1− .	بلغة **Java** – public int orderRoom (int type, int nights) بلغة **C#** – public int OrderRoom (int type, int nights)
تُعيد العمليّة مصفوفة بِكِبَر 3 تظهر فيها المجاميع الكلّيّة لمبالغ الدَّخْل من كلّ الغرف في كلّ واحد من الطوابق في الفندق. في المؤشّر **0** يظهر المجموع الكلّيّ للدَّخْل في الطابق **1**، وفي المؤشّر **1** يظهر المجموع الكلّيّ للدَّخْل في الطابق **2** ، وفي المؤشّر **2** يظهر المجموع الكلّيّ للدَّخْل في الطابق **3** .	بلغة **Java** – public int [] floorIncome () بلغة **C#** – public int [] FloorIncome ()

(1) طبِّق العمليّة orderRoom بلغة Java أو العمليّة OrderRoom بلغة C# ، في الفئة **Hostel** .

(2) طبِّق العمليّة floorIncome بلغة Java أو العمليّة FloorIncome بلغة C# ، في الفئة **Hostel** .

يجب استعمال العمليّة التي كتبتَها في البند "أ" .

/ يتبع في صفحة 8 /

الفصل الثاني (50 درجة)

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

أجب عن اثنين من الأسئلة 4-6 (لكلّ سؤال – 25 درجة).

4. أ. اكتب عمليّة خارجيّة isExist بلغة Java أو IsExist بلغة C# . تتلقّى العمليّة عددًا num – من نمط صحيح بين 0 وَ 9 (بما في ذلك 0 وَ 9)، وراصّة stk – من نمط صحيح. تُعيد العمليّة true إذا كان في الراصّة عدد رقم آحاده يساوي العدد num ، خلاف ذلك – تُعيد false .
افترِضْ أنّ الأعداد في الراصّة stk ليست سالبة.

مثال: بالنسبة لِـ num يساوي 8 والراصّة stk التي أمامك:

162	← رأس الراصّة
251	
568	
77	

تُعيد العمليّة true ، لأنّه يوجد في الراصّة العدد 568 الذي رقم آحاده هو 8 .
ملاحظة: يجب الحفاظ على مبنى الراصّة مع نهاية العمليّة.

ب. من أجل حلّ هذا البند فقط، بإمكانك استعمال العمليّة التي أمامك بدون الحاجة لتطبيقها:

وصف العمليّة	عنوان العمليّة
تتلقّى العمليّة راصّة من نمط صحيح وتُعيد نسخة دقيقة للراصّة بدون تغيير الراصّة الأصليّة.	**بلغة Java** – public static Stack <Integer> clone (Stack <Integer> s) **بلغة C#** – public static Stack <int> Clone (Stack <int> s)

نُعرِّف: الرقم الهامّ في العدد هو الرقم الموجود في أقصى اليسار فيه.
مثلاً الرقم 3 هو الرقم الهامّ في العدد 32 ، والرقم 5 هو الرقم الهامّ في العدد 541 .

اكتب عمليّة خارجيّة allExist بلغة Java أو AllExist بلغة C# ، تتلقّى راصّة stk ليست فارغة، من نمط صحيح. تُعيد العمليّة true إذا كانت جميع الأرقام الهامّة في الأعداد التي في الراصّة تظهر في رقم الآحاد في أعداد ما في الراصّة، خلاف ذلك – تُعيد false .
افترِضْ أنّ الأعداد في الراصّة stk ليست سالبة.

(انتبه: تكملة السؤال في الصفحة التالية.)
/يتبع في صفحة 9/

مثال: بالنسبة للراصّة stk التي أمامك:

122	← رأس الراصّة
251	
565	
12334	
28	
7	

تُعيد العمليّة true ، لأنّ جميع الأرقام الهامّة في الأعداد التي في الراصّة – 7 ، 5 ، 2 ، 1 – تظهر في رقم الآحاد في أعداد في الراصّة.

مثال آخر: بالنسبة للراصّة stk التي أمامك:

1223	← رأس الراصّة
245	
521	
12334	

تُعيد العمليّة false ، لأنّه من بين الأرقام الهامّة في الأعداد التي في الراصّة – 5 ، 2 ، 1 – الرقم 2 لا يظهر في رقم الآحاد في أيّ عدد من الأعداد التي في الراصّة.

ملاحظة: بإمكانك استعمال العمليّة التي كتبتَها في البند "أ".

/ يتبع في صفحة 10 /

5. تحضيرًا لمسابقة عَدْو الماراثون عُرِّفت الفئة **Competitor** التي تمثّل متنافِسًا أنهى المسار.

توجد للفئة ثلاث صفات:

- minutes – عدد الدقائق التي احتاجها المتنافِس لإنهاء المسار (عدد الدقائق ليس محدودًا حتّى 60)، من نمط صحيح.
- seconds – عدد الثواني التي احتاجها المتنافِس لإنهاء المسار (عدد الثواني هو حتّى 59 (بما في ذلك 59))، من نمط صحيح.
- name – اسم المتنافِس من نمط نصّ.

افترِضْ أنّه تمّ لكلّ صفة تعريف عمليّتَي get وَ set بلغة Java وعمليّتَي Get وَ Set بلغة C# .

بالإضافة إلى ذلك، عُرِّفت فئة باسم **Race** تجمع كلّ المتنافِسين الذين أنهوا المسار في مجموعة. عدد الذين أنهوا المسار ليس معروفًا.

افترِضْ أنّه لا يوجد متنافِسان أنهيا المسار في زمن متساوٍ.

أمامك واجهة تطبيق جزئيّة للفئة **Race** ، مكتوبة بلغة Java وبلغة C# :

التعقيدات	وصف العمليّة	عنوان العمليّة
O(n)	تتلقّى العمليّة كائنًا من نمط **Competitor** وتُضيفه إلى المجموعة. انتبه: العمليّة التالية لواجهة التطبيق (rank بلغة Java أو Rank بلغة C#) ومتطلّبات تعقيداتها لها صلة بالطريقة التي يُطبِّقون بها هذه العمليّة.	بلغة **Java** – public void add (Competitor x) بلغة **C#** – public void Add (Competitor x)
O(n)	تتلقّى العمليّة تدريجًا وتُعيد اسم المتنافِس، الذي له هذا التدريج في المجموعة. إرشاد: 1 هو المتنافِس الذي أنهى المسار في أقصر زمن، وَ 2 هو المتنافِس الذي أنهى المسار في ثاني أقصر زمن وهكذا. افترِضْ أنّه يوجد متنافِس بالتدريج المطلوب. ملاحظة: لا تَقُم بمحو حدود من المجموعة.	بلغة **Java** – public String rank (int x) بلغة **C#** – public string Rank (int x)

(انتبه: تكملة السؤال في الصفحة التالية.)
/ يتبع في صفحة 11 /

انتبه:

- لكلّ عمليّة توجد توجيهات تعقيدات زمن تشغيل.
- n هو عدد المتنافِسين في المجموعة.
- **يجب الإيفاء بمتطلّبات التعقيدات.**

بالنسبة للبندين " أ – ب"، عليك استعمال مبنى معطيات ملائم يفي بمتطلّبات السؤال.
بإمكانك استعمال أيّ مبنى معطيات تعلَّمتَه.
بإمكانك كتابة عمليّات في الفئة **Competitor**.

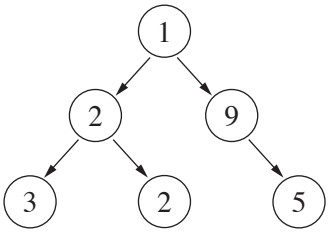
أ. اكتب صفات الفئة **Race**.

ب. طبِّق عمليّات الفئة **Race** التي تظهر في واجهة التطبيق التي في السؤال.

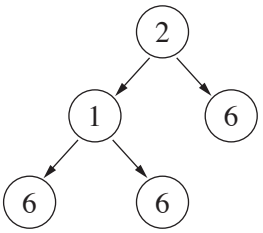
/ يتبع في صفحة 12 /

6. نُعرِّف: "شجرة أعداد" هي شجرة بيناريّة من نمط صحيح، تحوي كلّ عقدة فيها رقمًا بين 1 وَ 9 (بما في ذلك 1 وَ 9)، وكلّ مسار في الشجرة من الجذر إلى الورقة يمثّل عددًا: الورقة تمثّل رقم الآحاد، والمستوى الذي فوقها يمثّل رقم العشرات وهكذا حتّى جذر الشجرة.

مثال: في شجرة الأعداد التي أمامك ممثَّلة الأعداد: 195 ، 122 ، 123 (في مسارات في الشجرة من اليسار إلى اليمين).



مثال آخر: في شجرة الأعداد التي أمامك ممثَّلة الأعداد: 26 ، 216 ، 216 (في مسارات في الشجرة من اليسار إلى اليمين).



اكتب عمليّة خارجيّة printAll بلغة Java أو PrintAll بلغة C# . تتلقّى العمليّة شجرة أعداد tree من نمط صحيح وتطبع جميع الأعداد التي تمثّلها المسارات في الشجرة.

إذا كانت tree هي null لا تطبع العمليّة شيئًا.

ملاحظة: ليست هناك أهمّيّة لترتيب طباعة الأعداد.

/ يتبع في صفحة 13 /

الفصل الثالث (25 درجة)

في هذا الفصل أسئلة في أربعة مسارات:
أنظمة حاسوب وأسمبلي، في الصفحتين 13-15.
مقدّمة لبحث الأداءات، في الصفحات 16-20.
موديلات حسابيّة، في الصفحة 21 .
برمجة موجّهة كائنات بلغة Java ، في الصفحات 22-25؛ برمجة موجّهة كائنات بلغة C# ، في الصفحات 26-29.

أجب عن سؤال واحد في المسار الذي تعلّمتَه.

أنظمة حاسوب وأسمبلي

إذا تعلّمتَ هذا المسار، أجب عن سؤال واحد من السؤالين 7-8 (25 درجة).

7. في هذا السؤال ثلاثة بنود "أ – جـ". لا توجد علاقة بين البنود. أجب عن جميعها.

أ. أمامك قطعة برنامج بلغة أسمبلي.

```
    MOV BX, 50H
    MOV CL, 8
L1: MOV AX, [BX]
    ROL  AX, CL
    MOV [BX],  AX
    ADD BX, 2
    CMP BX, 57H
    JBE L1
```

أمامك خارطة خلايا الذاكرة من 50H وحتّى 57H ، قبل تنفيذ قطعة البرنامج:

عنوان الخليّة	50H	51H	52H	53H	54H	55H	56H	57H
محتوى الخليّة	11H	22H	33H	44H	55H	66H	77H	88H

ملاحظة: انتبه أنّ هناك فرقًا بين نقل معطيات من خلايا ذاكرة إلى خازن 8 بِتّات وبين نقل معطيات من خلايا ذاكرة إلى خازن 16 بِتًّا.

(1) تتبّع بواسطة جدول متابعة تنفيذ قطعة البرنامج. يجب أن يشمل جدول المتابعة عمودًا لكلّ واحد من الخوازن التي تظهر في قطعة البرنامج. بالإضافة إلى ذلك يجب رسم خارطة ذاكرة ملائمة.

(2) اشرح ما الذي تُنفِّذه قطعة البرنامج.

(انتبه: تكملة السؤال في الصفحة التالية.) / يتبع في صفحة 14 /

ب. أمامك قطعة برنامج مكتوبة بلغة Java وبلغة C# . اكتب قطعة ملائمة بلغة أسمبلي.

```
while ( a > 0 || b <= c )
{
    a = b + c ;
    c -- ;
}
```

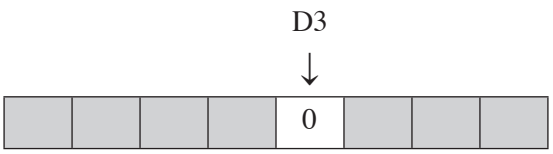
افترِضْ أنّ المتغيّرات a ، b ، c موجَّهة (signed) ومخزونة في الخوازن AX, BX, CX بالتلاؤم.

جـ. أمامك قولان بالنسبة للخازن AH . بالنسبة لكلّ واحد من القولين، حدِّد إذا كان صحيحًا أم غير صحيح. إذا كان القول صحيحًا علّل إجابتك، وإذا كان القول غير صحيح، عوِّض في الخازن AH عددًا يُثبِت أنّ هذا القول لا يتحقّق.

i. بافتراض أنّه في البِتّ D3 (البِتّ الرابع من الجهة اليمنى) للخازن AH مخزونة القيمة 0 ، فإنّ الأمرين 1 وَ 2 اللذين أمامك يُنفِّذان نفس العمليّة. أي أنّ قيمة الخازن AH مطابقة بعد تشغيل الأمرين.

الأمر 1	الأمر 2
AND AH, 11110111b	SUB AH, 8

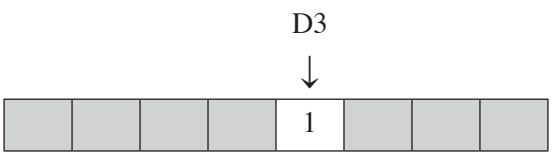
استعِن بالرسم التوضيحيّ الذي أمامك:



ii. بافتراض أنّه في البِتّ D3 (البِتّ الرابع من الجهة اليمنى) للخازن AH مخزونة القيمة 1، فإنّ الأمرين 1 وَ 2 اللذين أمامك يُنفِّذان نفس العمليّة. أي أنّ قيمة الخازن AH مطابقة بعد تشغيل الأمرين.

الأمر 1	الأمر 2
AND AH, 11110111b	SUB AH, 8

استعِن بالرسم التوضيحيّ الذي أمامك:



/ يتبع في صفحة 15 /

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

أ. أمامك قطعة برنامج بلغة أسمبلي.

```
MOV AX, 620H
SUB AH, 76H
```

حدِّد ما هي قِيَم الأعلام SF ، CF ، ZF بعد تنفيذ قطعة البرنامج. اشرح تحديدك.

ب. في مقطع المعطيات، عُرِّف المعطى الذي أمامك:

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

اكتب برنامجًا فرعيًّا باسم TEST يتلقّى عددين صحيحين عن طريق راصّة. العدد الأوّل الذي يُمرَّر في الراصّة هو قيمة الحدّ الذي يظهر في المصفوفة ARR ، والعدد الثاني الذي يُمرَّر في الراصّة هو مؤشّر نفس الحدّ.
إذا كان من بداية المصفوفة وحتّى المؤشّر الذي مُرِّر (لا يشمل المؤشّر) توجد على الأقلّ قيمة واحدة تساوي قيمة الحدّ الذي مُرِّر، يَخزِن البرنامج الفرعيّ الرقم 1 في الخازن AL ، خلاف ذلك – يَخزِن 0 في الخازن AL.

مثال: في المصفوفة ARR التي أمامك تُمرَّر القيمة 5 والمؤشّر 4 .

index	0	1	2	3	**4**	5	6	7	8	9
value	3	16	0	3	**5**	3	5	16	5	0

في هذا المثال، يَخزِن البرنامج الفرعيّ الرقم 0 في الخازن AL ، لأنّه من بداية المصفوفة وحتّى المؤشّر 4 ، القيمة 5 ليست قائمة.

مثال آخر: في المصفوفة ARR التي أمامك تُمرَّر القيمة 5 والمؤشّر 8 .

index	0	1	2	3	4	5	6	7	**8**	9
value	3	16	0	3	5	3	5	16	**5**	0

في هذا المثال، يَخزِن البرنامج الفرعيّ الرقم 1 في الخازن AL ، لأنّه من بداية المصفوفة وحتّى المؤشّر 8 ، القيمة 5 قائمة.

/ يتبع في صفحة 16 /

مقدّمة لبحث الأداءات

إذا تعلّمتَ هذا المسار، أجب عن سؤال واحد من السؤالين 9-10 (25 درجة) .

9. معطاة مسألة تخطيط خطّيّ :

$$\max \{z = ax_1 + 10x_2\}$$

تخضع للاضطرارات التالية :

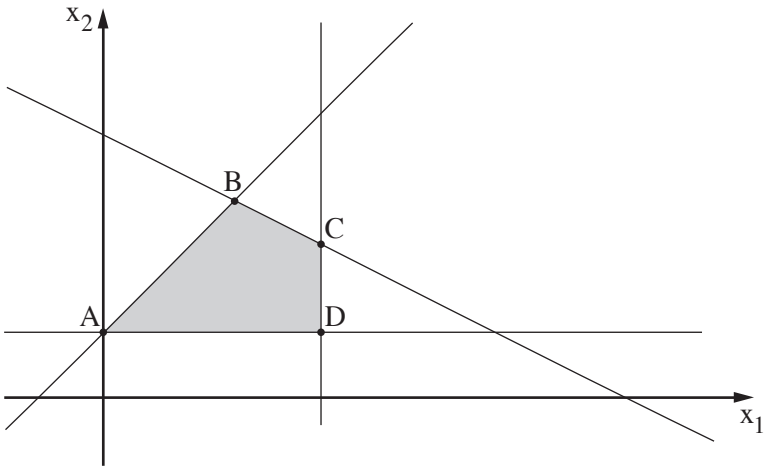
(1) $x_1 + 3 \geq x_2$

(2) $x_2 \geq 3$

(3) $-0.5x_1 + 12 \geq x_2$

(4) $x_1 \leq 10$

أمامك رسم لمجال الحلول الممكنة للمسألة المعطاة .



(انتبه : تكملة السؤال في الصفحة التالية .)

/ يتبع في صفحة 17 /

أ. جد إحداثيّات النقاط A ، B ، C ، D .

ب. على طول القطعة BC قيمة دالّة الهدف متطابقة. احسب a .

جـ. أمامك أربعة أقوال:

i. يوجد فقط حلّ أمثل واحد.

ii. توجد لانهاية من الحلول المثلى.

iii. الحلّ الأمثل ليس محصورًا.

iv. لا يوجد حلّ أمثل.

حدِّد أيّ قول من الأقوال i-iv صحيح. انسخه إلى دفترك، وعلّل تحديدك.

د. يُلغون الاضطرار الثاني $x_2 \geq 3$.

(1) ارسم مجال الحلول الممكنة من جديد.

(2) هل تَغَيَّرَ الحلّ الأمثل بعد هذا التغيير؟ علّل تحديدك.

(3) يُغيِّرون أيضًا دالّة الهدف إلى نهاية صغرى $\min\{z = ax_1 + 10x_2\}$.
هل تَغَيَّر الحلّ الأمثل بعد هذا التغيير؟ علّل تحديدك.

/ يتبع في صفحة 18 /

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

أ. **(1)** في الجدول الذي أمامك معطى حلّ لمسألة نقل. هل هذا الحلّ هو حلّ أساسيّ ممكن حسب طريقة الزاوية الشماليّة – الغربيّة؟ علّل إجابتك.

المصادر	الغايات				العرض
	A	B	C	D	
1	5 (100)	3	8 (100)	7 (100)	300
2	9	5 (300)	10 (100)	12 (100)	500
3	8	6	7	3 (200)	200
الطلب	100	300	200	400	

(انتبه: تكملة البند "أ" للسؤال في الصفحة التالية.)

/ يتبع في صفحة 19 /

(2) في الجدول الذي أمامك معطى حلّ أساسيّ ممكن نَتَج بعد k تكرارات بالنسبة لمسألة النقل، ومعطاة قيمة v_2 .

u_i	**العرض**	**الغايات**			**المصادر**
		3	2	1	
	100	8	3 (100)	10	A
	200	6	7 (100)	9 (100)	B
	400	4 (350)	9	18 (50)	C
	300	8	6 (300)	11	D
		350	500	150	**الطلب**
			0		v_j

i. انسخ الجدول إلى دفترك، وأكمل فيه القيَم v_3 ، v_1 ، u_4 ، u_3 ، u_2 ، u_1 .

ii. اشرح لماذا الحلّ ليس أمثل.

iii. عليك تنفيذ تكرار إضافيّ، أي التكرار $k+1$. ما هو المتغيّر الذي يخرج من القاعدة في هذا التكرار؟

iv. ارسم في دفترك جدولاً جديدًا، واكتب فيه الحلّ الذي نَتَج بعد هذا التكرار.

v. هل انخفضت التكلفة الكلّيّة لمسألة النقل؟

(انتبه: البند "ب" للسؤال في الصفحة التالية.)

/ يتبع في صفحة 20 /

ب. أمامك 6 مدن a ، b ، c ، d ، e ، f مرتبطة بشبكة شوارع. التكلفة الكلّيّة للسفر في الشارع مكوَّنة من مركِّبين: تكلفة الوقود ورسوم السفر.

أمامك مصفوفة مجاوَرات لتكلفة الوقود بالشواكل للسفر بين كلّ مدينتين.

	a	**b**	**c**	**d**	**e**	**f**
a	0	6	3	∞	∞	∞
b	6	0	7	2	11	∞
c	3	7	0	10	2	2
d	∞	2	10	0	8	2
e	∞	11	2	8	0	6
f	∞	∞	2	2	6	0

أمامك مصفوفة مجاوَرات لرسوم السفر بالشواكل للسفر بين كلّ مدينتين.

	a	**b**	**c**	**d**	**e**	**f**
a	0	4	20	∞	∞	∞
b	4	0	2	9	1	∞
c	20	2	0	10	13	5
d	∞	9	10	0	3	13
e	∞	1	13	3	0	3
f	∞	∞	5	13	3	0

جد أرخص مسارات للسفر حسب ألغوريثم ديكسترا، من المدينة a إلى كلّ واحدة من المدن b ، c ، d ، e ، f . اكتب أقلّ تكلفة كلّيّة للسفر من المدينة a إلى كلّ واحدة من المدن الأخرى.

/ يتبع في صفحة 21 /

موديلات حسابيّة

إذا تعلّمتَ هذا المسار، أجب عن سؤال واحد من السؤالين 11-12 (25 درجة) .

11. ابنِ آلة تيورينچ تتلقّى كمُدخَل الأعداد x ، y ، z من بداية الشريط (الأعداد ليست سالبة) .
x هو العدد الأوّل (من بداية الشريط) وَ y هو الثاني وَ z هو الثالث .
كلّ عدد مكتوب بصورة أونارّية .
الإشارة # تفصل بين كلّ عدد وآخر .
مثال : إذا تلقّى المُدخَل 2 بالنسبة لِـ x وَ 3 بالنسبة لِـ y وَ 1 بالنسبة لِـ z ، يبدو الشريط على النحو التالي :

⊢	1	1	#	1	1	1	#	1	△	△	△	....

مثال آخر : إذا تلقّى المُدخَل 0 بالنسبة لِـ x وَ 1 بالنسبة لِـ y وَ 0 بالنسبة لِـ z ، يبدو الشريط على النحو التالي :

⊢	#	1	#	△	△	△	....

تحسب الآلة قيمة الدالّة التي أمامك :

$$f(x,y,z)=\begin{cases} y+z & x=0 \\ x+y & x>0 \text{ وكذلك } x \text{ زوجيّ} \\ x & x \text{ فرديّ} \end{cases}$$

يُكتَب المُخرَج على الشريط في مكان ما كقيمة أونارّية بين إشارتَي $.

12. أ. ابنِ أوتومات راصّة بالنسبة لِلُّغة L_1 فوق الأبجديّة {a,b,c} مكوَّنًا من تسلسلات من الصورة $a^n b^k c^n$ بحيث n هو عدد فرديّ وباقي قسمة k على ثلاثة هو واحد . بين كلّ تسلسلين يفصل الحرف b .
مثالان لكلمتين تتبعان لِلُّغة L_1 :

abbbbcbaaabccc, abc

أمثلة لكلمات لا تتبع لِلُّغة L_1 :

abbc – لأنّ عدد المرّات التي يظهر فيها الحرف b هو 2 ، وباقي قسمة 2 على 3 هو 2 .
abcabc – لأنّ الحرف b لا يفصل بين التسلسلين .
abccc – لأنّ عدد المرّات التي يظهر فيها الحرف a لا يساوي عدد المرّات التي يظهر فيها الحرف c .
aabcc – لأنّ عدد المرّات التي يظهر فيها الحرف a والحرف c هو زوجيّ .

ب. معطاة اللغة L_2 فوق الأبجديّة {a,b,c}

$$L_2=\{a^k b^m c^x \mid 0\le k<5,\ 0\le m<5, 0\le x\}$$

نُعرِّف $L_3=L_2\cap L_1$.
اكتب جميع الكلمات في اللغة L_3 .

/ يتبع في صفحة 22 /

برمجة موجّهة كائنات بلغة Java

إذا تعلّمتَ هذا المسار وتكتب بلغة Java، أجب عن سؤال واحد من السؤالين 13-14 (25 درجة) .

13. معطاة الفئات : **A ، B ، C ، D ، E**

```
public class A
{
    protected int x ;
    public A ()
    {
        this.x = 9 ;
        System.out.println ( "A. x =  " +this.x ) ;
    }
    public A ( int x )
    {
        this.x = x ;
        System.out.println ( "A. x =  " + this.x ) ;
    }
    public int getX () { return this.x ; }
    public int foo () { return this.x ; }
}

public class B extends A
{
    public B () { super () ; }
    public B ( int x ) { super ( x ) ; }
    public int foo () { return this.x +1 ; }
}

public class C extends B
{
    public C () { super () ; }
    public C ( int x ) { super ( x ) ; }
    public int foo () { return this.x + 2 ; }
    public int bar () { return this.x ; }
}
```

```
public class D extends C
{
    public D ()
    {
        super () ;
        this.x ++ ;
        System.out.println ( "D. x = " + this.x ) ;
    }
    public D ( int x )
    {
        super ( x ) ;
        System.out.println ( "D. x = " + this.x ) ;
    }
    public D ( int x, int y )
    {
        super ();
        this.x = this.x + x + y;
        System.out.println ( "D. x = " + this.x ) ;
    }
    public int foo () { return this.x − 1 ; }
}

public class E extends C
{
    public E () { super () ; }
    public int bar () { return this.x +1 ; }
}
```

(انتبه : تكملة السؤال في الصفحة التالية .)

/ يتبع في صفحة 23 /

أ. ارسم مخطّطًا هرميًّا بين الفئات A ، B ، C ، D ، E . يجب الإشارة إلى توريث من الفئة بواسطة السهم ⟵ .

ب. أمامك عنوان العمليّة:

```
public static int getType ( Object m )
```

تُعيد العمليّة 1 إذا كان m من نمط A ، وَ 2 إذا كان m من نمط B ، وَ 3 إذا كان m من نمط C ، وَ 4 إذا كان m من نمط D ، وَ 5 إذا كان m من نمط E .
طبِّق العمليّة.
إرشاد: من أجل فحص نوع الكائن، يجب الاستعانة بالعمليّات foo ، bar ، getX .
لا تَستعمِل العمليّة instanceof وعمليّات الفئة Object ، ولا تُغيِّر الفئات A ، B ، C ، D ، E .
افترِضْ أنّ m يتبع لإحدى الفئات A ، B ، C ، D ، E وأنّه ليس null .

جـ. أمامك الفئة **Tester** :

```
public class Tester

{
    public static void main ( String [] args )

    {
        A a1 = new B () ;
        A a2 = new E () ;
        A a3 = new D () ;
        A a4 = new D ( 5 ) ;
        A a5 = new D ( 3, 7 ) ;
    }
}
```

اكتب مُخرَج العمليّة main .

/ يتبع في صفحة 24 /

14. في شركة الاتّصالات "آذان للمستقبل" يوجد 4 أنواع عاملين. كلّ عامل يتبع لنوع واحد فقط.

عامل عاديّ – عامل عاديّ في الشركة.

عضو لجنة – عامل عضو في لجنة العمّال.

فنّيّ – عامل مسؤول عن عدد من الحواسيب في الشركة (كلّ فنّيّ مسؤول عن عدد مختلف من الحواسيب).

مسؤول عن مشروع – عامل يُدير مشاريع خاصّة في الشركة. وهو مسؤول عن 15 عاملاً منهم عاملون عاديّون وأعضاء لجنة وفنّيّون.

تُتَّخذ القرارات من خلال تصويت يشارك فيه جميع عاملي الشركة. يُجرى التصويت بواسطة نقاط.
لكلّ عامل في الشركة يوجد عدد نقاط مختلف، حسب نوعه.
أمامك جدول فيه معطيات عن عدد النقاط التي توجد لكلّ عامل حسب نوعه:

نوع العامل	عدد النقاط
عامل عاديّ	يحصل على 4 نقاط لكونه عاملاً في الشركة، وبالإضافة إلى ذلك يحصل على نقطة أخرى عن كلّ سنة أقدميّة. مثال: للعامل العاديّ الذي يعمل 5 سنوات في الشركة توجد **9** نقاط: 4 نقاط لكونه عاملاً في الشركة وَ 5 نقاط أخرى – نقطة عن كلّ سنة أقدميّة في الشركة.
عضو لجنة	يحصل على ضعف عدد النقاط التي يحصل عليها العامل العاديّ، وبالإضافة إلى ذلك يحصل على نقطتين عن كلّ سنة يكون فيها عضو لجنة. مثال: لعضو لجنة يعمل 5 سنوات في الشركة، ومنها 3 سنوات كعضو لجنة توجد **24** نقطة: 18 نقطة – ضعف العامل العاديّ الذي يعمل 5 سنوات في الشركة. وَ 6 نقاط أخرى – نقطتان عن كلّ سنة كعضو لجنة.
فنّيّ	يحصل على نقاط كالعامل العاديّ، وبالإضافة إلى ذلك يحصل على نقطة عن كلّ حاسوب يقع ضمن مسؤوليّته. مثال: للفنّيّ الذي يعمل 5 سنوات في الشركة ومسؤول عن 4 حواسيب توجد **13** نقطة: 9 نقاط – كما يحصل العامل العاديّ الذي يعمل 5 سنوات في الشركة، وَ 4 نقاط أخرى – نقطة عن كلّ حاسوب يقع تحت مسؤوليّته.
مسؤول عن مشروع	يحصل على نقاط كالعامل العاديّ، وبالإضافة إلى ذلك يحصل على عدد نقاط يساوي العدد الكلّيّ لجميع نقاط العاملين الـ15 المسؤول عنهم. مثال: للمسؤول عن مشروع الذي يعمل 5 سنوات في الشركة، والعدد الكلّيّ لجميع نقاط العاملين الـ15 المسؤول عنهم هو 80، يحصل على **89** نقطة: 9 نقاط – كما يحصل العامل العاديّ الذي يعمل 5 سنوات في الشركة، وَ 80 نقطة أخرى – العدد الكلّيّ لنقاط العاملين الـ15 المسؤول عنهم.

(انتبه: تكملة السؤال في الصفحة التالية.) /يتبع في صفحة 25/

يرغبون في الشركة في بناء منظومة محوسَبة تعرض عدد نقاط كلّ عامل في الشركة. لهذا الغرض، عرَّفوا 4 فئات:
Employee – عامل عاديّ، **UnionMember** – عضو لجنة، **Technician** – فنّيّ،
Supervisor – مسؤول عن مشروع.
كلّ عامل يتبع لفئة واحدة فقط.
يجب حلّ البنود التي أمامك بالطريقة الأكثر ملاءَمة لمبادئ البرمجة الموجّهة كائنات.

أ. ارسم مخطّطًا هرميًّا بين الفئات. يجب الإشارة إلى توريث من الفئة بواسطة السهم ◄——
وإلى احتواء بواسطة الإشارة ◆—— .

ب. بالنسبة لكلّ واحدة من الفئات، اكتب عنوان الفئة وصفاتها والعمليّة getScore () التي تُعيد عدد النقاط في التصويت. لا حاجة لكتابة عمليّة بنائيّة وعمليّتَي get وَ set .
إرشاد: المعطيات الثابتة لعدد النقاط ليست جزءًا من صفات الفئات، ويجب استعمالها فقط في كتابة العمليّة getScore .

جـ. طُرِح اقتراح بتغيير اسم الشركة إلى "اتّصالات مستقبليّة". جميع العاملين العاديّين والفنّيّين أيّدوا الاقتراح، وجميع أعضاء اللجنة والمسؤولين عن المشاريع عارضوه. فقط إذا كان العدد الكلّيّ للنقاط التي تؤيّد الاقتراح أكبر من العدد الكلّيّ للنقاط التي تعارض الاقتراح – يُقبَل الاقتراح، خلاف ذلك لا يُقبَل.
معطى عنوان عمليّة تتلقّى مصفوفة جميع العاملين وتُعيد true إذا قُبل الاقتراح، خلاف ذلك – تُعيد false .

public static boolean isAccepted (Object [] arr)

طبِّق العمليّة.

/ يتبع في صفحة 26 /

برمجة موجّهة كائنات بلغة C#

إذا تعلّمتَ هذا المسار وتكتب بلغة C#، أجب عن سؤال واحد من السؤالين 15-16 (25 درجة) .

15. معطاة الفئات: **A ، B ، C ، D ، E**

```
public class A
{
    protected int x ;
    public A ()
    {
        this.x = 9 ;
        Console.WriteLine ( "A. x =  " +this.x ) ;
    }
    public A ( int x )
    {
        this.x = x ;
        Console.WriteLine ( "A. x =  " + this.x ) ;
    }
    public int GetX () { return this.x ; }
    public virtual int Foo () { return this.x ; }
}

public class B : A
{
    public B () : base () {}
    public B ( int x ) : base ( x ) {}
    public override int Foo () { return this.x +1 ; }
}

public class C : B
{
    public C () : base () {}
    public C ( int x ) : base ( x ) {}
    public override int Foo () { return this.x +2 ; }
    public virtual int Bar () { return this.x ; }
}
```

```
public class D : C
{
    public D () : base ()
    {
        this.x ++ ;
        Console.WriteLine ( "D. x = " + this.x ) ;
    }
    public D ( int x ) : base ( x )
    {
        Console.WriteLine ( "D. x = " + this.x ) ;
    }
    public D ( int x, int y ) : base ()
    {
        this.x = this.x + x + y;
        Console.WriteLine ( "D. x = " + this.x ) ;
    }
    public override int Foo () { return this.x − 1 ; }
}

public class E : C
{
    public E () : base () {}
    public override int Bar () { return this.x +1 ; }
}
```

(انتبه: تكملة السؤال في الصفحة التالية .)

/ يتبع في صفحة 27 /

أ. ارسم مخطّطًا هرميًّا بين الفئات E ، D ، C ، B ، A . يجب الإشارة إلى توريث من الفئة بواسطة السهم ◄—— .

ب. أمامك عنوان العمليّة:

```
public static int GetType ( Object m )
```

تُعيد العمليّة 1 إذا كان m من نمط A ، وَ 2 إذا كان m من نمط B ، وَ 3 إذا كان m من نمط C ، وَ 4 إذا كان m من نمط D ، وَ 5 إذا كان m من نمط E .
طبِّق العمليّة.
إرشاد: من أجل فحص نوع الكائن، يجب الاستعانة بالعمليّات Foo ، Bar ، GetX .
لا تَستعمِل العمليّتين is وَ as وعمليّات الفئة Object ، ولا تُغيِّر الفئات E ، D ، C ، B ، A .
افترِضْ أنّ m يتبع لإحدى الفئات E ، D ، C ، B ، A وأنّه ليس null .

جـ. أمامك الفئة **Tester** :

```
public class Tester

{
    public static void Main ( string [] args )

    {
        A a1 = new B () ;
        A a2 = new E () ;
        A a3 = new D () ;
        A a4 = new D ( 5 ) ;
        A a5 = new D ( 3, 7 ) ;
    }
}
```

اكتب مُخرَج العمليّة Main .

/ يتبع في صفحة 28 /

16. في شركة الاتّصالات "آذان للمستقبل" يوجد 4 أنواع عاملين. كلّ عامل يتبع لنوع واحد فقط.

عامل عاديّ – عامل عاديّ في الشركة.

عضو لجنة – عامل عضو في لجنة العمّال.

فنّيّ – عامل مسؤول عن عدد من الحواسيب في الشركة (كلّ فنّيّ مسؤول عن عدد مختلف من الحواسيب).

مسؤول عن مشروع – عامل يُدير مشاريع خاصّة في الشركة. وهو مسؤول عن 15 عاملاً منهم عاملون عاديّون وأعضاء لجنة وفنّيّون.

تُتَّخذ القرارات من خلال تصويت يشارك فيه جميع عاملي الشركة. يُجرى التصويت بواسطة نقاط.
لكلّ عامل في الشركة يوجد عدد نقاط مختلف، حسب نوعه.
أمامك جدول فيه معطيات عن عدد النقاط التي توجد لكلّ عامل حسب نوعه:

نوع العامل	عدد النقاط
عامل عاديّ	يحصل على 4 نقاط لكونه عاملاً في الشركة، وبالإضافة إلى ذلك يحصل على نقطة أخرى عن كلّ سنة أقدميّة. مثال: للعامل العاديّ الذي يعمل 5 سنوات في الشركة توجد **9** نقاط: 4 نقاط لكونه عاملاً في الشركة وَ 5 نقاط أخرى – نقطة عن كلّ سنة أقدميّة في الشركة.
عضو لجنة	يحصل على ضعف عدد النقاط التي يحصل عليها العامل العاديّ، وبالإضافة إلى ذلك يحصل على نقطتين عن كلّ سنة يكون فيها عضو لجنة. مثال: لعضو لجنة يعمل 5 سنوات في الشركة، ومنها 3 سنوات كعضو لجنة توجد **24** نقطة: 18 نقطة – ضعف العامل العاديّ الذي يعمل 5 سنوات في الشركة. وَ 6 نقاط أخرى – نقطتان عن كلّ سنة كعضو لجنة.
فنّيّ	يحصل على نقاط كالعامل العاديّ، وبالإضافة إلى ذلك يحصل على نقطة عن كلّ حاسوب يقع ضمن مسؤوليّته. مثال: للفنّيّ الذي يعمل 5 سنوات في الشركة ومسؤول عن 4 حواسيب توجد **13** نقطة: 9 نقاط – كما يحصل العامل العاديّ الذي يعمل 5 سنوات في الشركة، وَ 4 نقاط أخرى – نقطة عن كلّ حاسوب يقع تحت مسؤوليّته.
مسؤول عن مشروع	يحصل على نقاط كالعامل العاديّ، وبالإضافة إلى ذلك يحصل على عدد نقاط يساوي العدد الكلّيّ لجميع نقاط العاملين الـ15 المسؤول عنهم. مثال: للمسؤول عن مشروع الذي يعمل 5 سنوات في الشركة، والعدد الكلّيّ لجميع نقاط العاملين الـ15 المسؤول عنهم هو 80، يحصل على **89** نقطة: 9 نقاط – كما يحصل العامل العاديّ الذي يعمل 5 سنوات في الشركة، وَ 80 نقطة أخرى – العدد الكلّيّ لنقاط العاملين الـ15 المسؤول عنهم.

(انتبه: تكملة السؤال في الصفحة التالية.) / يتبع في صفحة 29 /

يرغبون في الشركة في بناء منظومة محوسَبة تعرض عدد نقاط كلّ عامل في الشركة. لهذا الغرض، عرَّفوا 4 فئات:
Employee – عامل عاديّ، **UnionMember** – عضو لجنة، **Technician** – فنّيّ،
Supervisor – مسؤول عن مشروع.
كلّ عامل يتبع لفئة واحدة فقط.
يجب حلّ البنود التي أمامك بالطريقة الأكثر ملاءَمة لمبادئ البرمجة الموجّهة كائنات.

أ. ارسم مخطّطًا هرميًّا بين الفئات. يجب الإشارة إلى توريث من الفئة بواسطة السهم ◄——
وإلى احتواء بواسطة الإشارة ◆—— .

ب. بالنسبة لكلّ واحدة من الفئات، اكتب عنوان الفئة وصفاتها والعمليّة GetScore () التي تُعيد عدد النقاط في التصويت. لا حاجة لكتابة عمليّة بنائيّة وعمليّتَي Get وَ Set .
إرشاد: المعطيات الثابتة لعدد النقاط ليست جزءًا من صفات الفئات، ويجب استعمالها فقط في كتابة العمليّة GetScore .

جـ. طُرِح اقتراح بتغيير اسم الشركة إلى "اتّصالات مستقبليّة". جميع العاملين العاديّين والفنّيّين أيّدوا الاقتراح، وجميع أعضاء اللجنة والمسؤولين عن المشاريع عارضوه. فقط إذا كان العدد الكلّيّ للنقاط التي تؤيّد الاقتراح أكبر من العدد الكلّيّ للنقاط التي تعارض الاقتراح – يُقبَل الاقتراح، خلاف ذلك لا يُقبَل.
معطى عنوان عمليّة تتلقّى مصفوفة جميع العاملين وتُعيد true إذا قُبل الاقتراح، خلاف ذلك – تُعيد false .

public static bool IsAccepted (Object [] arr)

طبِّق العمليّة.

בהצלחה!
نتمنّى لك النّجاح!