מדינת ישראל
משרד החינוך

סוג הבחינה: בגרות
מועד הבחינה: קיץ תשפ"ה, 2025
מספר השאלון: 899371
תרגום לערבית (2)

دولة إسرائيل
وزارة التّربية والتّعليم

نوع الامتحان: بچروت
موعد الامتحان: صيف 2025
رقم النّموذج: 899371
ترجمة إلى العربيّة (2)

מדעי המחשב

הוראות

א. משך הבחינה: שעתיים וחצי.

ב. מבנה השאלון ומפתח ההערכה:
בשאלון זה שני פרקים.
פרק ראשון — (20x2) — 40 נק'
פרק שני — (30x2) — 60 נק'
סך הכול — 100 נק'

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

ד. הוראה מיוחדת:
את כל התוכניות שיש לכתוב בשפת מחשב כתבו בשפה אחת בלבד — Java או C#.

הערה: לא יוּרדוּ נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.

علم الحاسوب

تعليمات

أ. مدّة الامتحان: ساعتان ونصف.

ب. مبنى النّموذج وتوزيع الدّرجات:
في هذا النّموذج فصلان.
الفصل الأوّل – (20x2) – 40 درجة
الفصل الثّاني – (30x2) – 60 درجة
المجموع – 100 درجة

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

د. تعليمة خاصّة:
اكتبوا جميع البرامج الّتي يجب كتابتها بلغة حاسوب، بلغة واحدة فقط – Java أو C#.

ملاحظة: لن تُخصم درجات إذا كتبتم في البرامج حرفًا كبيرًا بدلًا من حرف صغير أو بالعكس.

يجب الكتابة في دفتر الامتحان فقط. يجب كتابة "مسوّدة" في بداية كلّ صفحة تُستعمَل مسوّدة.
كتابة أيّة مسوّدة على أوراق خارج دفتر الامتحان قد تسبّب إلغاء الامتحان.

الأسئلة في هذا النّموذج ترد بصيغة الجمع، ورغم ذلك يجب على كلّ طالبة وطالب الإجابة عنها بشكل فرديّ.

ב ה צ ל ח ה !

نتمنّى لكم النّجاح!

الأسئلة

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

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

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

الفصل الأوّل (40 درجة)

أجيبوا عن اثنين من الأسئلة 1-3. (لكلّ سؤال – 20 درجة.)

1. أمامكم العمليّة what بلغة Java أو What بلغة C#:

Java

```
public static int what (int[] arr)
{
    int y = 0;
    int x = arr[0];
    for (int i = 0; i < arr.length; i++)
    {
        y += arr[i];
        x = Math.max (x, y);
        y = Math.max (y, 0);
    }
    return x;
}
```

C#

```
public static int What (int[] arr)
{
    int y = 0;
    int x = arr[0];
    for (int i = 0 ;i < arr.Length; i++)
    {
        y += arr[i];
        x = Math.Max (x, y);
        y = Math.Max (y, 0);
    }
    return x;
}
```

معطاة المصفوفة arr:

	0	1	2	3	4	5
arr	3	4	−8	10	−6	8

تتبَّعوا، بواسطة جدول متابعة، العمليّة what(arr) بلغة Java أو What(arr) بلغة C#، واكتبوا ماذا تُعيد العمليّة.
يجب أن يشمل جدول المتابعة عمودًا لكلّ واحد من المتغيّرات التالية: x ، y ، i.

/ يتبع في صفحة 3 /

2. طبِّقوا العمليّة arrange بلغة Java أو Arrange بلغة C# :

Java – public static void arrange (int[] arr)

C# – public static void Arrange (int[] arr)

في المصفوفة arr المتلقّاة، توجد أعداد موجبة وأعداد سالبة.
العمليّة تنقل إلى بداية المصفوفة arr جميع الأعداد الموجبة (بما فيها 0). وَتَضَعُ بعدها جميع الأعداد السالبة.
ترتيب الأعداد الموجبة (فيما بينها) يبقى بنفس الترتيب الذي كان قبل العمليّة، وكذلك ترتيب الأعداد السالبة (فيما بينها) يبقى بنفس الترتيب الذي كان قبل العمليّة.

مثال:
بالنسبة للمصفوفة arr التي أمامكم:

	0	1	2	3	4	5	6	7
arr	7	−2	−5	61	4	−1	0	33

المصفوفة arr تبدو على هذا النحو في نهاية العمليّة:

	0	1	2	3	4	5	6	7
arr	7	61	4	0	33	−2	−5	−1

الشرح: نقلت العمليّة الأعداد الموجبة إلى بداية المصفوفة، وحافظت على الترتيب الأصليّ للأعداد الموجبة وللأعداد السالبة.

/ يتبع في صفحة 4 /

3. عدد موجب – num يُسمّى "عددًا نرجسيًّا" إذا تحقَّق الشرط الذي أمامكم:
يرفعون كلّ واحد من أرقام num بِأُسّ عدد الأرقام الموجودة في num (طول العدد num)،
يجمعون كلّ النتائج، والمجموع الناتج مساوٍ لِـ num .

مثال لِـ "عدد نرجسيّ": العدد 407 ، لأنّه عندما نرفع كلّ واحد من أرقامه بالأُسّ 3 (الذي هو عدد الأرقام في العدد)،
نحصل على 407 : $4^3 + 0^3 + 7^3 = 64 + 0 + 343 = 407$

مثال لعدد **ليس** "عددًا نرجسيًّا": العدد 58 ، لأنّه عندما نرفع كلّ واحد من أرقامه بالأُسّ 2 (الذي هو عدد الأرقام في العدد)،
نحصل على 89 : $5^2 + 8^2 = 25 + 64 = 89$.

أ. اكتبوا عمليّة خارجيّة باسم isNarc بلغة Java أو IsNarc بلغة C# ، تتلقّى عددًا من نمط صحيح – num
أكبر من 0 . تُعيد العمليّة true إذا كان العدد نرجسيًّا، خلاف ذلك تُعيد العمليّة false .

ب. اكتبوا عمليّة خارجيّة باسم theNarc بلغة Java أو TheNarc بلغة C# ، تتلقّى عددًا من نمط صحيح – n
أكبر من 0 . تطبع العمليّة جميع الأعداد النرجسيّة الموجودة من 1 حتّى n (بما في ذلك 1 وَ n).
ملاحظة: يجب استعمال العمليّة التي كتبتموها في البند "أ" .

/ يتبع في صفحة 5 /

الفصل الثاني (60 درجة)

أجيبوا عن اثنين من الأسئلة 4-6 . (لكلّ سؤال – 30 درجة .)

4. تتبع عدّة بلدات للمجلس الإقليميّ "عوز" . يُوفِّر المجلس للأولاد الذين يسكنون في هذه البلدات خدمات توصيل من البيت إلى المدرسة وإعادة إلى البيت . التوصيل من البيت إلى المدرسة يُسمّى "توصيل ذهاب" وتوصيل العودة من المدرسة إلى البيت يُسمّى "توصيل إياب" .

معطاة الفئة **Transport** – توصيل، ولها أربع صفات :

- name – اسم البلدة التي يتمّ التوصيل بالنسبة لها، من نمط نصّ .
- toSchool – نوع التوصيل . بالنسبة لِـ "توصيل ذهاب" ، قيمة الصفة هي true ، وبالنسبة لِـ "توصيل إياب" القيمة هي false .
- num – عدد الطلّاب في التوصيل – عدد من نمط صحيح من 1 حتّى 50 (بما في ذلك 1 وَ 50) .
- day – رقم من نمط صحيح من 1 حتّى 6 (بما في ذلك 1 وَ 6)، يمثِّل اليوم في الأسبوع الذي يتمّ فيه التوصيل (الأحد – الجمعة بالتلاؤم) .

افترِضوا أنّه توجد عمليّات get/Get لصفات الفئة .

أ. **(1)** اكتبوا عمليّة بنائيّة في الفئة **Transport** ، تتلقّى اسم البلدة التي يتمّ التوصيل بالنسبة لها – name ، وعدد الطلّاب في التوصيل – num، واليوم في الأسبوع الذي يتمّ فيه التوصيل – day .
العمليّة تبتدئ صفات الفئة . في نوع التوصيل (toSchool) تُكوِّن العمليّة "توصيل ذهاب" .
افترِضوا أنّ البارامترات التي يتمّ تلقّيها سليمة .

(2) اكتبوا عمليّة بنائيّة في الفئة **Transport** تتلقّى فقط عدد الطلّاب في التوصيل – num .
العمليّة تبتدئ صفات الفئة بحيث يُتلقَّى "توصيل إياب" في البلدة Aviv يوم الأربعاء ويكون في هذا التوصيل num طلّاب .
ملاحظة : إذا لم يكن بارامتر عدد الطلّاب (num) سليمًا، تتلقّى الصفة 1 .

ب. معطاة مصفوفة – arr ، من نمط **Transport** ، تحوي التوصيلات التي تمّت خلال الأسبوع . المصفوفة ليست مرتَّبة حسب ترتيب أيًّا كان، ولا توجد فيها قيَم null .

(1) اكتبوا عمليّة خارجيّة باسم dayReport بلغة Java أو DayReport بلغة C# ، تتلقّى المصفوفة arr ، ورقمًا من نمط صحيح – day يُشير إلى اليوم في الأسبوع، وبارامترًا بوليانيًّا – forward .
إذا كان forward هو true، تُعيد العمليّة العدد الكلّيّ للطلّاب الذين سافروا في "توصيلات الذهاب" في اليوم – day . خلاف ذلك (إذا كان forward هو false)، تُعيد العمليّة العدد الكلّيّ للطلّاب الذين سافروا في "توصيلات الإياب" في اليوم – day .
افترِضوا أنّ البارامتر day المتلقَّى سليم (من 1 حتّى 6) .

(2) اكتبوا عمليّة خارجيّة باسم moreForward بلغة Java أو MoreForward بلغة C# ، تتلقّى المصفوفة arr .
تطبع العمليّة أرقام الأيّام في الأسبوع التي كان فيها عدد الطلّاب الذين سافروا في "توصيلات الذهاب" أكبر من عدد الطلّاب الذين سافروا في "توصيلات الإياب" .
ملاحظة : يمكن الاستعانة بالعمليّة التي كتبتموها في البند الفرعيّ "ب (1)" .

/ يتبع في صفحة 6 /

5. معطاة الفئة **User** – مُستخدِم في شبكة تواصل اجتماعيّ، ولها ثلاث صفات:

- name – اسم المُستخدِم من نمط نصّ (يمكن أن يكون عدّة أشخاص بنفس الاسم).
- id – رقم هويّة من نمط صحيح (رقم مميِّز).
- friends – مصفوفة من نمط صحيح، فيها أرقام هويّة أصدقاء المُستخدِم (كلّ رقم يظهر مرّة واحدة فقط).
  كِبَر المصفوفة هو حسب عدد أصدقاء المُستخدِم.

افترِضوا أنّه توجد عمليّات get/Get لصفات الفئة.

أ. اكتبوا في الفئة **User** عمليّة داخليّة باسم mutual بلغة Java أو Mutual بلغة C# ، تتلقّى مُستخدِمًا (User) آخر – other . تُعيد العمليّة عدد الأصدقاء المشترَكين للمُستخدِمَيْن.

ب. معطاة الفئة **SocialNetwork** – شبكة تواصل اجتماعيّ، ولها صفة واحدة:

- users – مصفوفة من نمط **User**.

المصفوفة ليست مرتَّبة حسب ترتيب أيًّا كان، وكلّ مُستخدِم يظهر فيها مرّة واحدة فقط، وليس فيها قيَم null .

اكتبوا في الفئة **SocialNetwork** عمليّة داخليّة باسم exactOne بلغة Java أو ExactOne بلغة C# ، تتلقّى مُستخدِمًا (User) – other ، لا يظهر في المصفوفة. تُعيد العمليّة true إذا كان في المصفوفة مُستخدِم واحد على الأقلّ يوجد له ولِـother بالضبط صديق واحد مشترَك. خلاف ذلك، تُعيد false .
ملاحظة: يمكن الاستعانة بالعمليّة التي كتبتموها في البند "أ".

/ يتبع في صفحة 7 /

6. عدد موجب – num يُسمّى "عددًا كاملًا"، إذا تحقّق الشرط الذي أمامكم:
يجمعون جميع القواسم الموجبة والصحيحة لِـ num (الأعداد التي إذا قسمنا num عليها نحصل على نتيجة بدون باقٍ)، باستثنائه، والمجموع الناتج مساوٍ للعدد الأصليّ.

مثال لعدد كامل: العدد 6، لأنّ قواسمه هي 1، 2، 3، ومجموعها هو 6.
مثال إضافيّ لعدد كامل: العدد 28، لأنّ قواسمه هي 1، 2، 4، 7، 14، ومجموعها هو 28.
مثال لعدد **ليس** عددًا كاملًا: العدد 8، لأنّ قواسمه هي 1، 2، 4، ومجموعها هو 7.

أ. **(1)** اكتبوا عمليّة خارجيّة باسم isPerfect بلغة Java أو IsPerfect بلغة C#، تتلقّى عددًا صحيحًا – num أكبر من 2. تُعيد العمليّة true إذا كان عددًا كاملًا، خلاف ذلك تُعيد false.

(2) اكتبوا عمليّة خارجيّة باسم thePerfects بلغة Java أو ThePerfects بلغة C#، تتلقّى عددَيْن صحيحَيْن أكبر من 2: high، low. تطبع العمليّة جميع الأعداد الكاملة التي بين هذَيْن العددَيْن (بما في ذلك العددان).
افترِضوا أنّ low أصغر من high أو مساوٍ له.
ملاحظة: يمكن الاستعانة بالعمليّة التي كتبتموها في البند الفرعيّ "(1)".
مثال: بالنسبة لِـ low=4 وَ high=10 تطبع العمليّة 6.
الشرح: من العدد 4 وحتّى 10، العدد 6 فقط هو عدد كامل (العدد 4 ليس عددًا كاملًا، لأنّ مجموع قواسمه 1، 2 هو 3، والعدد 5 ليس عددًا كاملًا، لأنّه يقسم فقط على العدد 1، وهكذا).

ب. هناك فرضيّة بأنّه لا يوجد عدد فرديّ هو عدد كامل.
اكتبوا عمليّة خارجيّة باسم noOdd بلغة Java أو NoOdd بلغة C#، تُعيد true إذا كان بالفعل لا يوجد عدد كامل هو عدد فرديّ من بين جميع الأعداد 3 حتّى 999,999 (بما في ذلك 3 وَ 999,999). خلاف ذلك تُعيد العمليّة false.
ملاحظة: يمكن الاستعانة بالعمليّة التي كتبتموها في البند الفرعيّ "أ (1)".

בהצלחה!
نتمنّى لكم النّجاح!