

סוג הבחינה: מבחן מתכונת
מבחן: מתכונת 2
מספר השאלון: 899371
שם התלמיד/ה: _____

מדעי המחשב

יסודות מדעי המחשב – 1-2 יחידות לימוד

הוראות

- א. משך הבחינה: שעה וחצי.
- ב. מבנה השאלון ומפתח ההערכה: בשאלון זה שני פרקים.
פרק ראשון – (2×20) – 40 נקודות
פרק שני – (2×30) – 60 נקודות
סך הכול – 100 נקודות
- 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 יורדו נקודות אם תכתבו בתוכניות אות גדולה במקום אות קטנה או להפך.
יש לכתוב במחברת הבחינה בלבד. יש לרשום "טיוטה" בראש כל עמוד המשמש טיוטה.
כתיבת טיוטה בדפים שאינם במחברת הבחינה עלולה לגרום לפסילת הבחינה.
השאלות בשאלון זה מנוסחות בלשון רבים, אף על פי כן על כל תלמידה וכל תלמיד להשיב עליהן באופן אישי.
בהצלחה!
השאלות
בשאלון זה שני פרקים.
יש לענות על שאלות משני הפרקים, לפי ההוראות בכל פרק.
הערה: בכל שאלה שנדרשת בה קליטה, אין צורך לבדוק את תקינות הקלט.
לפותרים בשפת Java: בכל שאלה שנדרשת בה קליטה, הניחו שבתוכנית כתובה ההוראה:
`Scanner scan = new Scanner(System.in);`

השאלות

פרק ראשון (40 נקודות)

ענו על שתיים מן השאלות 1-3 (לכל שאלה - 20 נקודות).

1. נתון מערך arr, המכיל מספרים שלמים:

6	5	4	3	2	1	0	
5	6	9	10	3	7	4	arr

נתון קטע תוכנית, הכתוב בשפות Java ו-C#:

Java

Java

```
int count = 0;
for (int i = 0; i < arr.length; i++)
{
    if (arr[i] % 2 == 0)
    {
        count++;
        arr[i] = arr[i] / 2;
    }
    else
        arr[i] = arr[i] * count;
}
```

C#

C#

```
int count = 0;
for (int i = 0; i < arr.Length; i++)
{
    if (arr[i] % 2 == 0)
    {
        count++;
        arr[i] = arr[i] / 2;
    }
    else
        arr[i] = arr[i] * count;
}
```

א. עקבו בעזרת טבלת מעקב אחר ביצוע קטע התוכנית, ורשמו את תוכן המערך arr לאחר ביצוע קטע התוכנית.

בטבלת המעקב יש לכלול עמודה עבור כל אחד מן המשתנים האלה: count, arr[i], i.

ב. כתבו דוגמה למערך arr בגודל 5, שאף אחד מן התאים שלו אינו מכיל 0, ושלאחר ביצוע קטע התוכנית יהיו בו בדיוק שלושה תאים המכילים 0.

2. (20 נקודות)

חיסונים נשמרים במקרר מיוחד, ובכל שעה נמדדת הטמפרטורה במקרר (במעלות צלזיוס, מספר שלם). הפרמטר low הוא מספר שלם המייצג את הטמפרטורה הנמוכה ביותר המותרת במקרר. הפרמטר high הוא מספר שלם המייצג את הטמפרטורה הגבוהה ביותר המותרת במקרר ($low < high$).
א. כתבו פעולה חיצונית ששמה `getDeviation` בשפת Java או `GetDeviation` בשפת #C המקבלת מספר שלם - `temp`, המייצג טמפרטורה שנמדדה במקרר, ואת הפרמטרים `low` ו-`high`. הפעולה תחזיר את הסטייה של הטמפרטורה מן הטווח המותר, על פי התנאים האלה:
 - אם הטמפרטורה נמוכה מ-`low`, הפעולה תחזיר בכמה מעלות היא נמוכה מ-`low`.
 - אם הטמפרטורה גבוהה מ-`high`, הפעולה תחזיר בכמה מעלות היא גבוהה מ-`high`.
 - אם שני התנאים שלעיל אינם מתקיימים (הטמפרטורה בטווח המותר, כולל הגבולות), הפעולה תחזיר 0.

דוגמה: בעבור `low = 2`, `high = 8`: בעבור `temp = 11` תחזיר 3, בעבור `temp = -1` תחזיר 3, בעבור `temp = 8` תחזיר 0.

ב. המערך `temps` הוא מערך מטיפוס שלם המכיל את הטמפרטורות שנמדדו במקרר במהלך משלוח של חיסונים, לפי סדר המדידות (כל תא במערך מציג מדידה אחרת).
 כתבו פעולה חיצונית ששמה `getAllDeviations` בשפת Java או `GetAllDeviations` בשפת #C המקבלת את המערך `temps` ואת הפרמטרים `low` ו-`high`. הפעולה תחזיר מערך חדש מטיפוס שלם בגודל זהה למערך `temps`, המציג בכל תא את הסטייה של המדידה שנמצאת באותו האינדקס במערך `temps` (על פי סעיף א).
דוגמה: בעבור `low = 2`, `high = 8`, והמערך `temps`:

אינדקס	0	1	2	3	4	5	6
temps	5	9	11	7	1	-1	8

יוחזר המערך שלהלן:

אינדקס	0	1	2	3	4	5	6
	0	1	3	0	1	3	0

הערות: - יש להשתמש בפעולה שכתבתם בסעיף א.
 - אין לשנות את המערך `temps`.

ג. משלוח חיסונים נפסל אם מתקיים בו לפחות אחד מן התנאים האלה:
 - יש מדידה שבה הסטייה גדולה מ-3 מעלות.
 - יש שתי מדידות רצופות (בשני תאים סמוכים במערך) שבשתייהן הטמפרטורה מחוץ לטווח המותר.

כתבו פעולה חיצונית ששמה `isSafe` בשפת Java או `IsSafe` בשפת #C, המקבלת את המערך `temps` ואת הפרמטרים `low` ו-`high`. הפעולה תחזיר `true` אם המשלוח לא נפסל, אחרת הפעולה תחזיר `false`.
דוגמה: בעבור המערך הנתון בסעיף ב הפעולה תחזיר `false`, כי במדידות באינדקסים 1 ו-2, שהן רצופות, הטמפרטורה מחוץ לטווח המותר (אף שאין במערך מדידה שבה הסטייה גדולה מ-3).
הערה: יש להשתמש בפעולה שכתבתם בסעיף ב.

"כרטיס הגרלה" הוא מערך מטיפוס שלם שבו כל תא מכיל מספר שלם בין 1 ל-37 (כולל), וכל המספרים בכרטיס שונים זה מזה.

ממלאים כרטיס הגרלה תא אחר תא, החל מתא 0, עד שכל התאים מלאים. בכל שלב:

i. מוגרל מספר שלם בין 1 ל-37.

ii. המספר שהוגרל מוצב בתא הבא שיש למלא.

הערה: אם המספר שהוגרל כבר מופיע באחד התאים שמולאו, ההגרלה לא תיחשב (המספר לא יוצב), ומגרילים מספר חדש עבור אותו התא.

כתבו פעולה חיצונית ששמה fillTicket בשפת Java או FillTicket בשפת #C המקבלת מערך מטיפוס שלם - arr (שגודלו בין 1 ל-37) וממלאת אותו כ"כרטיס הגרלה".

הפעולה תדפיס את ערכי התאים לפי הסדר לאחר המילוי (כל ערך בשורה חדשה).

כתבו פעולה חיצונית ששמה fillTicket בשפת Java או FillTicket בשפת #C המקבלת מערך מטיפוס שלם - arr (שגודלו בין 1 ל-37) וממלאת אותו כ"כרטיס הגרלה".

הפעולה תדפיס את ערכי התאים לפי הסדר לאחר המילוי (כל ערך בשורה חדשה).

פרק שני (60 נקודות)

ענו על שתיים מן השאלות 4-6 (לכל שאלה - 30 נקודות).

4. (30 נקודות)

חברה ליבוא קפה מקבלת שקים של פולי קפה ירוקים במכולות.

נתונה המחלקה Sack המייצגת שק של פולי קפה ולה שתי תכונות: ארץ המקור של הפולים - origin, ומשקל השק בקילוגרמים - weight (מספר ממשי חיובי).

Java

```
public class Sack{
    private String origin;
    private double weight;
}
```

בנוסף, נתונה המחלקה Container המייצגת מכולה ולה שתי תכונות: קוד המכולה - code, ומערך של עצמים sacks מטיפוס Sack בגודל מספר השקים במכולה.

Java

```
public class Container{
    private String code;
    private Sack[] sacks;
}
```

הניחו שקיימות פעולות get ו-set בשפת Java ופעולות Get ו-Set בשפת C# בעבור כל התכונות בשתי המחלקות.

לפניכם ממשק חלקי של המחלקה Container:

תיאור הפעולה	כותרת הפעולה
פעולה בונה המקבלת את קוד המכולה ואת מספר השקים שבה, ומאתחלת את התכונות. המערך sacks מאותחל למערך ריק בגודל num	public Container(String code, int num)
פעולה המחזירה את המשקל הכולל (בקילוגרמים) של כל השקים במכולה.	- #Java - public double totalWeight() C public double TotalWeight()
פעולה המקבלת משקל מרבי limit (בקילוגרמים). הפעולה מחזירה true אם המשקל הכולל של השקים במכולה גדול מ-limit, אחרת מחזירה false.	Java - public boolean isOverweight(double limit) C - public bool IsOverweight(double limit)

א. ממשו את הפעולה הבונה של המחלקה Container.

ב. ממשו את הפעולה isOverweight (בשפת C#: IsOverweight) במחלקה Container. ניתן להשתמש בפעולה totalWeight של המחלקה Container מבלי לממש אותה. הניחו כי איברי המערך sacks שונים מ-null.

ג. מכולה יכולה לצאת למשלוח אם המשקל הכולל של השקים בה אינו גדול מהמשקל המרבי המותר.

כתבו פעולה חיצונית ששמה printReady (בשפת C#: PrintReady) המקבלת מערך של עצמים port מטיפוס Container, המייצג את המכולות שבנמל, ומשקל מרבי מותר limit. הפעולה תדפיס את הקודים של כל המכולות שיכולות לצאת למשלוח. חובה להשתמש בפעולה isOverweight של המחלקה Container. הניחו כי איברי המערכים port ו-sacks שונים מ-null.

5. (30 נקודות)

רכבת מורכבת מקרונות המחוברים זה לזה בשורה. יש קרונות נוסעים, קרונות משא, קרונות מכל (לנוזלים) וקרונות אחרים (קטר, קרון מזנון וכו').

נתונה המחלקה **Wagon** - קרון, ולה שלוש תכונות:

· **number** - המספר הסידורי של הקרון, מספר שלם

· **seats** - מספר המושבים בקרון, מספר שלם (0 בקרון שאין בו מושבים)

· **cargo** - סוג המטען, מחרוזת (מחרוזת ריקה בקרון שאין בו מטען)

הניחו שיש פעולות **get/Set** ו-**set/Set** בעבור תכונות המחלקה. ניתן להשתמש בפעולות המחלקה מבלי לממש אותן.

נתון ממשק חלקי של המחלקה **Wagon**:

תיאור הפעולה	כותרת הפעולה
פעולה בונה המקבלת ערכים בעבור תכונות המחלקה	<code>public Wagon(int number, int seats, String cargo)</code>
פעולה המחזירה true אם הקרון הוא קרון נוסעים, אחרת היא מחזירה false	<code>#Java - public boolean isPassenger()</code> <code>C public bool IsPassenger() -</code>
פעולה המחזירה true אם הקרון הוא קרון משא, אחרת היא מחזירה false	<code>- #Java - public boolean isFreight()</code> <code>C public bool IsFreight()</code>
פעולה המחזירה true אם הקרון הוא קרון מכל, אחרת היא מחזירה false	<code>- #Java - public boolean isTanker()</code> <code>C public bool IsTanker()</code>
פעולה המחזירה true אם הקרון אינו קרון נוסעים, אינו קרון משא ואינו קרון מכל, אחרת היא מחזירה false	<code>- #Java - public boolean isOther()</code> <code>C public bool IsOther()</code>

נתונה המחלקה **Train** - רכבת, ולה תכונה אחת בלבד:

· **wagons** - מערך מטיפוס **Wagon** בגודל **N**, ובו קרונות הרכבת לפי סדרם.

1. רכבת נקראת **מאוזנת** אם ההפרש בין כל שניים מבין שלושת המספרים - מספר קרונות הנוסעים,

מספר קרונות המשא ומספר קרונות המכל - הוא לכל היותר 1.

לדוגמה, רכבת שיש בה 3 קרונות נוסעים, 2 קרונות משא ו-3 קרונות מכל היא מאוזנת. רכבת שיש

בה 3 קרונות נוסעים, קרון משא אחד ו-2 קרונות מכל אינה מאוזנת.

כתבו פעולה פנימית במחלקה **Train** ששמה **isBalanced** בשפת **Java** או **IsBalanced** בשפת **C#**.

הפעולה תחזיר **true** אם הרכבת מאוזנת, אחרת הפעולה תחזיר **false**.

שימו לב: הקרונות האחרים אינם משפיעים על התשובה.

2. לצורך ניקיון מחלקים את הרכבת **למקטעים** של **m** קרונות רצופים: הקרונות בתאים 0 עד **m-1** הם

המקטע הראשון, הקרונות בתאים **m** עד **2m-1** הם המקטע השני, וכן הלאה.

כתבו פעולה פנימית במחלקה **Train** ששמה **hasFullSection** בשפת **Java** או **HasFullSection** בשפת

C#, המקבלת מספר שלם **m**.

הפעולה תחזיר **true** אם קיים מקטע כך שכל הקרונות בו הם קרונות נוסעים. אחרת, הפעולה

תחזיר **false**.

הניחו כי **m** חיובי ו-**N** מתחלק ב-**m** ללא שארית.

לדוגמה, בעבור הרכבת:

7	6	5	4	3	2	1	0
קטר	מכל	מכל	נוסעים	נוסעים	נוסעים	נוסעים	משא

(4) hasFullSection תחזיר false, ו-(2) hasFullSection תחזיר true (התאים 2-3).

6. (30 נקודות)

מספר שלם חיובי - num ייקרא "מספר הרשד" אם num מתחלק ללא שארית בסכום הספרות שלו.
דוגמה למספר הרשד: המספר 18, כי סכום הספרות שלו הוא $9 = 8 + 1$, ו-18 מתחלק ב-9 ללא שארית.
דוגמה נוספת למספר הרשד: המספר 201, כי סכום הספרות שלו הוא 3, ו-201 מתחלק ב-3 ללא שארית ($201 = 3 \cdot 67$).
דוגמה למספר שאינו מספר הרשד: המספר 19, כי סכום הספרות שלו הוא 10, ו-19 אינו מתחלק ב-10 ללא שארית.

א. (1) כתבו פעולה חיצונית ששמה isHarshad בשפת Java או IsHarshad בשפת C#, המקבלת מספר שלם חיובי - num. הפעולה **מחזירה** true אם הוא מספר הרשד, אחרת היא מחזירה false.
 (2) כתבו פעולה חיצונית ששמה printHarshad בשפת Java או PrintHarshad בשפת C#, המקבלת שני מספרים שלמים חיוביים: low, high. הפעולה **תדפיס** את כל מספרי הרשד שבין שני מספרים אלה (כולל).

הניחו ש-low קטן מ-high או שווה לו.

הערה: אפשר להיעזר בפעולה שכתבתם בתת-סעיף (1).

דוגמה: בעבור low=10 ו-high=21 הפעולה תדפיס 10, 12, 18, 20, 21.

הסבר: המספר 10 הוא מספר הרשד (סכום הספרות 1), המספר 11 אינו מספר הרשד (סכום הספרות 2 ו-1 אי-זוגי), המספר 12 הוא מספר הרשד (סכום הספרות 3), וכן הלאה.

ב. **רצף** של מספרי הרשד הוא כמה מספרים עוקבים שכולם מספרי הרשד. למשל, 20, 21 הם רצף של שני מספרי הרשד.

יש השערה **שלא קיים** רצף של שישה מספרי הרשד עוקבים בין המספרים 10 עד 999,999 (כולל).

כתבו פעולה חיצונית ששמה noSixInRow בשפת Java או NoSixInRow בשפת C#, **המחזירה** true אם אכן מבין כל המספרים 10 עד 999,999 (כולל) אין שישה מספרי הרשד עוקבים. אחרת היא **מחזירה** false.

הערה: אפשר להיעזר בפעולה שכתבתם בתת-סעיף א(1).

כל הבגרויות במדעי המחשב עם פתרונות מלאים: www.guyai.co.il/bagrut/computer-science