

סוג הבחינה: מבחן מתכונת
מבחן: מתכונת 5
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. (20 נקודות)

נתון מערך `arr`, המכיל מספרים שלמים חיוביים.

6	5	4	3	2	1	0	
14	271	7	52	915	38	253	arr

לפניכם קטע תוכנית, הכתוב בשפות Java ו-#C.

Java

Java

```
int count=0;
for (int i=1; i<arr.length; i++)
{
    int last=arr[i-1]%10;
    int first=arr[i];
    while (first>=10)
        first=first/10;
    if (last==first)
        count++;
}
```

#C

C#

```
int count=0;
for (int i=1; i<arr.Length; i++)
{
    int last=arr[i-1]%10;
    int first=arr[i];
    while (first>=10)
        first=first/10;
    if (last==first)
        count++;
}
```

א. עקבו בעזרת טבלת מעקב אחר ביצוע קטע התוכנית עבור המערך `arr`, ורשמו את ערך המשתנה `count` לאחר ביצוע קטע התוכנית.

בטבלת המעקב יש לכלול עמודה עבור כל אחד מן המשתנים האלה: `first`, `last`, `i`, `count`.

ב. כתבו דוגמה למערך `arr` בגודל 5 המכיל מספרים שלמים חיוביים (גדולים מ-0) שבסיום קטע התוכנית ערכו של המשתנה `count` יישאר 0.

ג. כתבו דוגמה למערך `arr` בגודל 5 המכיל מספרים שלמים חיוביים (גדולים מ-0) שבסיום קטע התוכנית ערכו של המשתנה `count` יהיה שווה ל-4.

2.

בחממה לגידול ירקות מותקן חיישן המודד כמה פעמים ביום את אחוז הלחות באוויר. כל מדידה היא מספר שלם מ-0 ועד 100.

יום נחשב **יום תקין** בחממה אם מתקיימים כל התנאים האלה:

. ההפרש בין המדידה הגבוהה ביותר לבין המדידה הנמוכה ביותר באותו יום הוא **לכל היותר** 25.

. אין באותו יום שתי מדידות **רצופות** שבשתייהן הלחות נמוכה מ-40.

. **לפחות שליש** מן המדידות באותו יום הן 60 ומעלה.

דוגמה: ביום מסוים נעשו שש מדידות, לפי הסדר: 55, 62, 48, 70, 66, 51. ההפרש בין 70 ל-48 הוא 22, אין מדידות הנמוכות מ-40, ושלוש מתוך שש המדידות (62, 70, 66) הן 60 ומעלה – יותר משליש מהן. לכן זהו יום תקין.

דוגמה נוספת: ביום מסוים נעשו חמש מדידות, לפי הסדר: 45, 38, 39, 50, 60. המדידות 38 ו-39 הן רצופות ושתייהן נמוכות מ-40, ולכן זהו אינו יום תקין.

דוגמה נוספת: ביום מסוים נעשו ארבע מדידות, לפי הסדר: 72, 58, 45, 61. ההפרש בין 72 ל-45 הוא 27, ולכן זהו אינו יום תקין.

כתבו פעולה חיצונית ששמה goodDay בשפת Java או GoodDay בשפת C#, המקבלת מספר שלם - num.

num מציין את מספר המדידות שנעשו בחממה ביום מסוים. הניחו כי num גדול מ-0.

הפעולה תקלוט את num המדידות, לפי הסדר שבו נעשו.

הפעולה תחזיר true אם היום הוא יום תקין, אחרת תחזיר הפעולה false.

(20 נקודות)

3. (20 נקודות)

במחסן של חנות מקוונת כל מוצר מסומן באות אחת מן האותיות Z-A.

הזמנה של לקוח מיוצגת על ידי מחרוזת של אותיות המוצרים שהזמין. עובד מלקט עובר במחסן ואוסף מוצרים, והמוצרים שאסף מיוצגים על ידי מחרוזת ליקוט. בכל אחת מן המחרוזות כל אות יכולה להופיע **פעם אחת** לכל היותר, והאותיות אינן מסודרות בסדר כלשהו (המלקט אינו חייב לאסוף את המוצרים לפי סדר ההזמנה).

בסיום הליקוט כל מוצר מקבל אחד מן הסימונים האלה:

- "נאסף" – מוצר שמופיע בהזמנה ונאסף.

- "חסר" – מוצר שמופיע בהזמנה אך לא נאסף.

- "מיותר" – מוצר שנאסף אך אינו מופיע בהזמנה.

לדוגמה: בעבור ההזמנה "KDRWP" ומחרוזת הליקוט "PWXDQ":

שלושה מוצרים סומנו "נאסף" – P, W, D.

שני מוצרים סומנו "חסר" – R, K.

שני מוצרים סומנו "מיותר" – X, Q.

כתבו פעולה חיצונית ששמה `checkPicking` בשפת Java או `CheckPicking` בשפת C#, המקבלת שני פרמטרים:
 מחרוזת הזמנה - `order`, ומחרוזת ליקוט - `picked`.
 הניחו ששתי המחרוזות תקינות (כלומר מופיעות בהן רק האותיות Z-A וכל אות מופיעה בכל מחרוזת פעם אחת לכל היותר), ושמהרזות ההזמנה אינה ריקה.
 אם מחרוזת הליקוט ריקה (המלקט לא אסף דבר), הפעולה תדפיס הודעת שגיאה "Nothing Picked".
 אחרת, הפעולה תדפיס את מספר המוצרים שסומנו "נאסף", מספר המוצרים שסומנו "חסר" ומספר המוצרים שסומנו "מיותר" (כל הדפסה בשורה חדשה).

פרק שני (60 נקודות)

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

4. (30 נקודות)

נתונה המחלקה **PrintPlan** - מסלול הדפסה חודשי של סטודנט במכון ההדפסה של האוניברסיטה, ולה חמש תכונות:

- `studentId` - מספר הזיהוי של הסטודנט, מטיפוס שלם
 - `fee` - התשלום החודשי הקבוע על המסלול, מטיפוס שלם
 - `maxPages` - מספר עמודי השחור-לבן הכלולים במסלול בכל חודש, מטיפוס שלם
 - `maxColor` - מספר עמודי הצבע הכלולים במסלול בכל חודש, מטיפוס שלם
 - `extra` - הסכום הנוסף לתשלום על חריגה ממספר העמודים הכלולים במסלול, מטיפוס שלם
- תעריפי החריגה הם שקל אחד בעבור כל 5 עמודי שחור-לבן נוספים (רק קבוצות שלמות של 5 עמודים מחויבות; למשל, על חריגה של 27 עמודים משלמים 5 שקלים), ו-2 שקלים בעבור כל עמוד צבע נוסף. הניחו שקיימות פעולות `get/Get` לתכונות המחלקה.

א. (1) כתבו פעולה בונה במחלקה **PrintPlan**, המקבלת את מספר הזיהוי של הסטודנט - `studentId`, את התשלום החודשי - `fee`, את מספר עמודי השחור-לבן הכלולים במסלול - `maxPages`, ואת מספר עמודי הצבע הכלולים במסלול - `maxColor`. הפעולה מאתחלת את תכונות המחלקה. התכונה `extra` מאותחלת ל-0.

ב. (2) כתבו פעולה פנימית במחלקה **PrintPlan** ששמה `setExtra` בשפת Java או `SetExtra` בשפת C#, המקבלת את מספר עמודי השחור-לבן שהדפיס הסטודנט בפועל בחודש - `pages` (מטיפוס שלם), ואת מספר עמודי הצבע שהדפיס בפועל בחודש - `color` (מטיפוס שלם). הפעולה קובעת ערך לתכונה `extra` בהתאם לחריגה ממסלול ההדפסה (ככל שיש) ולתעריפי החריגה.

הערה: `extra` היא 0 לפני הפעולה.

ג. (1) נתון מערך מלא - `plans` מטיפוס **PrintPlan**, רבו נתוני המסלולים של סטודנטים שונים (לאחר שכבר חושב ועודכן הסכום הנוסף לתשלום בתכונה `extra`).

כתבו פעולה חיצונית ששמה `countHigh` בשפת Java או `CountHigh` בשפת C#, המקבלת את המערך `plans`, ומחזירה את מספר הסטודנטים שהסכום הנוסף שלהם (`extra`) גדול מן התשלום החודשי הקבוע שלהם (`fee`).

(2) כתבו פעולה חיצונית ששמה highStudents בשפת Java או HighStudents בשפת C#, המקבלת את המערך plans . הפעולה מחזירה מערך מטיפוס שלם ובו מספרי הזיהוי (studentId) של הסטודנטים שהסכום הנוסף שלהם גדול מן התשלום החודשי שלהם (גודל המערך הוא בדיוק כמספר הסטודנטים האלה). אין חשיבות לסדר ההופעה במערך.
הערה: חובה להיעזר בפעולה שכתבתם בסעיף ב(1).

5. (30 נקודות)

מועדון צפרות מתעד את מיני הציפורים שכל אחד מחבריו ראה בתצפיות.
נתונה המחלקה **Watcher** - צפר, ולה שלוש תכונות:
· name - שם הצפר מטיפוס מחרוזת (ייתכנו כמה צפרים שלהם אותו השם)
· id - מספר החבר במועדון מטיפוס שלם (מספר ייחודי)
· species - מערך מטיפוס שלם, ובו הקודים של מיני הציפורים שהצפר ראה (כל קוד מופיע פעם אחת בלבד). גודל המערך הוא בהתאם למספר המינים שהצפר ראה.
הניחו שקיימות פעולות get/Get לתכונות המחלקה.
א. כתבו במחלקה **Watcher** פעולה פנימית ששמה together בשפת Java או Together בשפת C#, המקבלת צפר (Watcher) אחר - other . הפעולה מחזירה את מספר מיני הציפורים **השונים** שראה לפחות אחד משני הצפרים.
דוגמה: אם הצפר הנוכחי ראה את המינים 4, 9, 15, 22 ו- other ראה את המינים 9, 22, 31, הפעולה תחזיר 5 (המינים 4, 9, 15, 22, 31).
ב. נתונה המחלקה **BirdClub** - מועדון צפרות, ולה תכונה אחת:
· members - מערך מטיפוס **Watcher**
המערך אינו ממורכז לפי סדר כלשהו, כל צפר מופיע בו פעם אחת בלבד, ואין בו ערכי null . המועדון מתכנן תחרות זוגות לצפר חדש. כתבו במחלקה **BirdClub** פעולה פנימית ששמה allReach בשפת Java או AllReach בשפת C#, המקבלת צפר (Watcher) - w, שאינו מופיע במערך, ומספר שלם - k . הפעולה מחזירה true אם לכל צפר במערך מתקיים: מספר מיני הציפורים השונים שראו הצפר ו- w יחד הוא k לפחות. אחרת הפעולה מחזירה false .
הערה: אפשר להיעזר בפעולה שכתבתם בסעיף א.

6. (30 נקודות)

שעון דיגיטלי מציג את השעה בתבנית hh:mm - שתי ספרות לשעה (00 עד 23) ושתי ספרות לדקות (00 עד 59).

שעת פלינדרום היא שעה שבה ארבע הספרות המוצגות נקראות אותו הדבר משמאל לימין ומימין לשמאל.

דוגמאות:

שעה	שעת פלינדרום?
13:31	כן
05:50	כן
22:22	כן
10:10	לא
12:34	לא

ביממה יש בדיוק 16 שעות פלינדרום.

נתונה המחלקה **Time** - שעה ביממה, ולה שתי תכונות:

Java

```
public class Time {
    private int hour ;
    private int minute ;
}
```

hour - מספר שלם מ-0 עד 23, minute - מספר שלם מ-0 עד 59.

הנח שקיימות פעולות get ו-set בשפת Java ופעולות Get ו-Set בשפת C# בעבור תכונות המחלקה. לפניך ממשק חלקי של המחלקה **Time** בשפות Java ו-C#. אפשר להשתמש בפעולות הממשק ללא צורך לממש אותן.

תיאור הפעולה	כותרת הפעולה
פעולה בונה המקבלת ערכים בעבור תכונות המחלקה.	public Time (int hour, int minute)
פעולה המחזירה true אם העצם הוא שעת פלינדרום, אחרת היא מחזירה false.	בשפת Java: public boolean isPalindrome () בשפת C#: public bool IsPalindrome ()
פעולה המחזירה true אם לעצם הנוכחי ולעצם other יש אותה שעה ואותן דקות, אחרת היא מחזירה false.	בשפת Java: public boolean sameAs (Time (other) בשפת C#: public bool SameAs (Time (other)

לפניך כותרת פעולה חיצונית בשפת :

Java

```
public static Time[] palindromeTimes (int n)
```

C#

```
public static Time[] PalindromeTimes (int n)
```

הפעולה `palindromeTimes / PalindromeTimes` בונה מערך ריק של עצמים מטיפוס `Time` בגודל n .
בעבור כל תא במערך, הפעולה מגרילה שעה בין 0 ל-23 (כולל) ודקות בין 0 ל-59 (כולל). אם השעה
שהוגרלה היא שעת פלינדרום וגם היא אינה נמצאת כבר באחד התאים הקודמים של המערך, מוכנס
לתא עצם שערכי התכונות שלו הם הערכים שהוגרלו. אחרת, הפעולה ממשיכה להגריל עד שמתקבלת
שעה כזאת, ואז מוכנס לתא עצם שערכי התכונות שלו הם הערכים שהוגרלו.
לאחר שהמערך מתמלא, הפעולה מחזירה את המערך.
הערה: הנחש n בין 1 ל-16 (כולל).
ממשו את הפעולה.

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