שאלה 1

סעיף א - 10 נק', 4 ל-1 ו-3 לכל חישוב סיבוכיות

נתונה תוכנית ה-C הבאה:

```
#include <stdio.h>

int bar(int m, int d)
{
    if(d <= 1) return 1;
    if(!(m%d)) return 0;
    return(bar(m, d-1));
}

void foo(int n)
{
    if(!n)
        return;
    if(bar(n, n-1)) printf("%d\n", n);
    foo(n-1);
}

int main ()
{
   int n;
   scanf("%d", &n);
   foo(n);

   return 0;
}
```

1. מה מבצעת התכנית ומה יהיה הפלט עבור הקלט 10?
2. מהי סיבוכיות הזמן ומהי סיבוכיות המקום של התוכנית כפונקציה של **n**?

התוכנית מדפיסה את המספרים הראשוניים הקטנים מ או שווים ל-n.
עבור הקלט 10 תדפיס התוכנית 7, 5, 3, 2, 1

סיבוכיות זמן: התוכנית קוראת ל-bar() על כל המספרים מ-n ועד ל-1, הסיבוכיות של bar() היא d כי יש d קריאות רקורסיביות לפונקציה. מכיוון ש-foo() מתחילה תמיד עם d=n-1, מתקבל:

$Time(foo()) = (n-1)+(n-2)+\ldots+3+2+1 = O(n^2)$

מבחינת סיבוכיות מקום - עומק הרקורסיה המקסימלי הוא n (קריאה ל-foo() ועוד n-1 קריאות ל-bar()), ולכן

$Space(foo()) = O(n)$

סעיף ב - 15 נק', 3 לכל שאלה

בכל אחד מהסעיפים הבאים מופיעות מספר שורות קוד. לכל קטע קוד, הקיפו בעיגול את התיאור המתאים:

א. **ללא שגיאות** – הקוד יתקמפל ללא כל שגיאה וירוץ ללא תקלות.
ב. **שגיאת זמן ריצה** – הקוד יתקמפל ללא שגיאות, אולם הוא **עלול** לבצע שגיאה בזמן הריצה.
ג. **שגיאת קומפילציה** – הקוד לא יעבור קומפילציה.

1.

```
char *p = "Oh, well";
p[2] = '!';
```

p מצביע לאזור בזכרון שיכול להיות read only, ולכן כתיבה אליו תגרום לשגיאת זמן ריצה.

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

2.

```
void foo(char bar[10]) {
  int i;
  for(i=0; i<10; i++)
    putchar(*(bar++));
}
```

אין בעיה – bar הוא מצביע, ולכן אפשר לשנות אותו.

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

3.

```
char *p = "%dx, %c%c";
printf(p, 10, 'c', 'u');
```

אין בעיה – printf מצפה לקבל מצביע מטיפוס *char וזה מה שאנחנו נותנים לו

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

4.

```
char s[5]={'H','e','l','l','o'};
int d=5.9999;

s[d] = 'O';
```

d יאותחל ל-5, אבל האינדקס 5 אינו חוקי במערך s שבו יש 5 איברים (אינדקסים 0 .. 4)

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה

5.

```
int a=1, *b=&a, **c=&b;

a = (c==&&a)? 2 : **c;
```

האופרטור & נותן כתובת של משתנה, אבל &&a אינו כתיבה חוקית (אין כזה דבר "כתובת של כתובת", כתובת חייבת להיות של משתנה!)
למעשה, כמו שאתם יודעים, && הוא אופרטור ב-C (AND לוגי) והשימוש בו כאן אינו נכון.

א. ללא שגיאות
ב. שגיאת זמן ריצה
ג. שגיאת קומפילציה