מבוא למדעי המחשב 202-1-101-1

סמסטר א' תשס"ד

מבחן מועד ב'

מר בועז בן-משה
פרופ' דניאל ברנד
פרופ' מיכאל קודיש
דר' יצחק רוזן

משך המבחן שעתיים וחצי – <u>לא תינתן הארכה</u>.
חומר עזר - אסור.
אין להשתמש במחשבון.

במבחן זה 4 שאלות, בניקוד המסתכם ב- 100 נקודות. ענו על כל השאלות.

אנא רשמו את תשובותיכם בדף התשובות בלבד. המחברת שקיבלתם היא **מחברת טיוטה ולא תימסר כלל לבדיקה.** בסיום הבחינה נשמור אך ורק את דף התשובות. כל שאר החומר יועבר לגריסה.
הקפידו לרשום בדף התשובות גם את **מספר הנבחן ומספר החדר בו אתם נבחנים**.

בשאלות התכנות, מספר השורות העומדות לרשותכם בדף התשובות רומז על אורך הקוד הנדרש. הקפידו על כתב יד ברור. תשובות מסורבלות או ארוכות מדי לא תזכינה בניקוד מלא. אין צורך להעתיק את שורות הקוד הנתונות בשאלון לדף התשובות.

בהצלחה!

<u>שאלה 1 (25 נקודות)</u>

השיטה הבאה מקבלת מערך grades של מספרים שלמים שערכם בין 0 ל- 100. על השיטה לחשב ולהחזיר מערך חדש hist, המייצג את היסטוגרמת הציונים באופן הבא:

הערך של hist[i] הוא מספר הפעמים שהציון i מופיע ב- grades.

לדוגמא:

מערך הקלט grades	**המערך hist לאחר סיום השיטה**
{90,90,50,100}	hist[90]=2, hist[50]=1, hist[100]=1 כל שאר האיברים הם 0
{0,100,0}	hist[0]=2, hist[100]=1 כל שאר האיברים הם 0

השלימו את קטע הקוד הבא כך שיבצע את הנדרש:

```
static int[] histogram(int[] grades) {
        // fill in
        return hist;
}
```

- ניתן להניח כי הפרמטר המתקבל הוא מערך שאינו null של מספרים שלמים בעל ערכים בין 0 ל- 100.

הפתרון:

```
int [] hist = new int[101];
for (int i=0; i<grades.length;i++) {
        hist[grades[i]]  += 1;
}
```

<u>שאלה 2 (24 נקודות)</u>

נזכר בהגדרות הבאות:
<u>הגדרה</u>: ביטוי postfix הוא מספר שלם (לשם פשטות נניח, כי הוא אי-שלילי)
או ביטוי מהצורה (e_1 e_2 op), באשר e_1 ו- e_2 ביטויי postfix ו- op אופארטור בינארי (לשם פשטות נניח + או *).
ביטוי infix הוא מספר שלם או ביטוי מהצורה (e_1 op e_2) באשר e_1 ו- e_2 ביטויי infix ו- op אופארטור בינארי.

<u>דוגמא</u>:
הביטויים הבאים <u>אינם</u> ביטויי postfix חוקיים:
3 + 4 , 2 3 4 * , 1 2 3 * + +

הביטויים הבאים <u>הינם</u> ביטויי postfix חוקיים:
3 , 3 4 + , 2 3 4 + * , 1 2 3 * 4 + +

הביטויים הבאים <u>הינם</u> ביטויי infix חוקיים: (הביטויים הנ"ל הם "תרגום" של ביטויי ה-postfix דלעיל).
3 , (3+4) , (2*(3+4)) , (1+((2*3)+4))

בשאלה זו אנו מעוניינים לכתוב פונקציה המתרגמת ביטויי postfix לביטויי infix.
לרשותכם המחלקה StackAsArray, המייצגת מחסנית בעלת בנאי ריק, וממשת את הממשק Stack :

```
public interface Stack{
        public void push(Object a);   // מוסיף איבר לראש המחסנית
        public Object pop();          // שולף ומחזיר את האיבר בראש המחסנית
                                      //  אם המחסנית ריקה נזרקת שגיאה
        public boolean isEmpty();     // מחזיר אמת אם ורק אם המחסנית ריקה
}
```

נתונה גם הפונקציה הסטטית print_infix(String exp), המקבלת ביטוי postfix (חוקי) ומדפיסה ביטוי infix תואם (מתורגם). הניחו שהמחרוזת exp שונה מ- null ומהווה ביטוי postfix חוקי.

```
  static void print_infix(String exp){
        System.out.println(infix(exp));
  }
```

להזכירכם: למחלקה StringTokenizer יש בנאי המקבל מחרוזת (המייצגת משפט). השיטה nextToken() מחזירה את "המילה" הבאה, בעוד שהשיטה hasMoreElements() מחזירה אמת אם ורק אם במשפט יש עדיין מילים שטרם "הגענו" עליהן.
לדוגמא, התוכנית הבאה:

```
StringTokenizer t = new StringTokenizer("1 2 *");
 while (t.hasMoreElements()) System.out.println(t.nextToken());
```

תדפיס:

```
1
2
*
```

<u>שאלה 2 - המשך</u>

<u>סעיף א' (17 נק')</u>

השלימו את הפונקציה הסטטית infix שמקבלת ביטוי postfix, ומחזירה ביטוי infix תואם (כמחרוזת).

הניחו שהמחרוזת exp שונה מ- null ומהווה **<u>ביטוי postfix חוקי.</u>**

```
static String infix(String exp) {
    Stack s = new StackAsArray();
    StringTokenizer t = new StringTokenizer(exp);
    while (t.hasMoreElements()){
        String token = t.nextToken();
        if (token.equals("+") | token.equals("*"))
```

```
            // השלימו את תשובתכם בדף התשובות
```

```
    } //while
    String ans = (String)s.pop();
    if(s.isEmpty()) return ans;
    return "ERROR";
}
```

<u>סעיף ב' (7 נק')</u>

כתבו מחדש את הפונקציה הסטטית print_infix(String exp) כך שאם המחרוזת exp הינה ביטוי postfix חוקי, היא <u>תדפיס</u> ביטוי infix תואם, אחרת תדפיס את המחרוזת "ERROR".

הניחו שהמחרוזת exp שונה מ- null, ומכיל אך ורק מספרים טבעיים ופעולות (+,*) וביניהן רווח.

ניתן בהחלט לענות על השאלה גם אם לא עניתם על סעיף א'.

רמז: מומלץ להשתמש ב Exception handling (טיפול בחריגים).

```
static void print_infix(String exp){
        String ans = null;
        // השלימו את תשובתכם בדף התשובות
        System.out.println(ans);
}
```

תשובה 2.1:

```
static String infix(String exp) {
   Stack s = new StackAsArray();
   StringTokenizer t = new StringTokenizer(exp);

   while (t.hasMoreElements()){
      String token = t.nextToken();
      if (token.equals("+") | token.equals("*")){
            String value2 = (String) s.pop();
            String value1 = (String) s.pop();
            s.push(new String("("+value1+token+value2+")"));
       }
       else s.push(token);
   }
   String ans = (String)s.pop();
   if(s.isEmpty()) return ans;
   return "ERROR";
}
```

תשובה 2.2

```
   static void print_infix(String exp){
            String ans = null;
            try { ans = infix(exp); }
            catch(Exception e) {ans = "ERROR";}
            System.out.println(ans);
   }
```

<u>שאלה 3 (20 נקודות)</u>

השיטה **int numberOfSubsets (int n, int size, int sum)** מחזירה עבור מספר שלם אי-שלילי n את מספר התת-קבוצות של הקבוצה {1,2,3,..,n} שמספר אבריהן הוא בדיוק size , וסכום אבריהן הוא בדיוק sum.

לדוגמא, הקריאה **numberOfSubsets (6, 2, 8)** תחזיר את הערך 2, שכן ישנן 2 תתי-קבוצה של {1,2,..,8} בהם שני האיברים מסתכמים ל-8, והן: {2,6},{3,5}.

השלימו את קטע הקוד הבא כך שיבצע את הנדרש.

ניתן להניח כי כל הפרמטרים בקריאה לשיטה הם אי-שליליים.

```
static int numberOfSubsets(int n, int size, int sum)
{
   if(n==0)
      if(size==0 & sum==0) return 1;
      else return 0;
   else
     // fill in
}// method numberOfSubsets
```

תשובה 3

```
//*****************************************************************
**
// SubsetsFixedSum.java
//
// Constructs all subsets of size k of {1,2,...,n} of sum s.
//*****************************************************************
**

public class SubsetsFixedSum
{
   public static void main (String[] args)
   {
      System.out.println("Subsets of size " + args[1] + " of sum " +
args[2]
                         + " of {1,...," + args[0] +"}");
      subsets(Integer.parseInt(args[0]), Integer.parseInt(args[1]),
Integer.parseInt(args[2]));
      System.out.println(numberOfSubsets(Integer.parseInt(args[0]),
Integer.parseInt(args[1]),
                                         Integer.parseInt(args[2])));
   }// method main

   static void subsets(int n, int size, int sum)
   {subsets(n, size, sum, "");}// method subsets

   static void subsets(int n, int size, int sum, String suffix)
   {
      if(size==0 && sum==0)
         System.out.println(suffix);
      else
         if(size>0)
         {
            subsets(n - 1, size - 1, sum - n, " " + n + suffix);
            if(size < n)
               subsets(n-1, size, sum, suffix);
         }
   }// method subsets

   static int numberOfSubsets(int n, int size, int sum)
   {
      if(n==0)
         if(size==0 & sum==0) return 1;
         else return 0;
      else return numberOfSubsets(n - 1, size - 1, sum - n)
                  + numberOfSubsets(n - 1, size, sum);
   }// method numberOfSubsets
}// class SubsetsFixedSum
```

שאלה 4 (31 נקודות)

בשאלה זו נתייחס לעץ בינארי שבקודקודיו יש ערכים שלמים (int). ניתן להניח שכל הערכים בעץ שונים זה מזה.

נתונות המחלקות עץ בינארי (BinaryTree) וצומת בעץ בינארי (BTN). (לכל צומת יש ערך שלם):

למחלקה עץ בינארי השיטות הבאות:

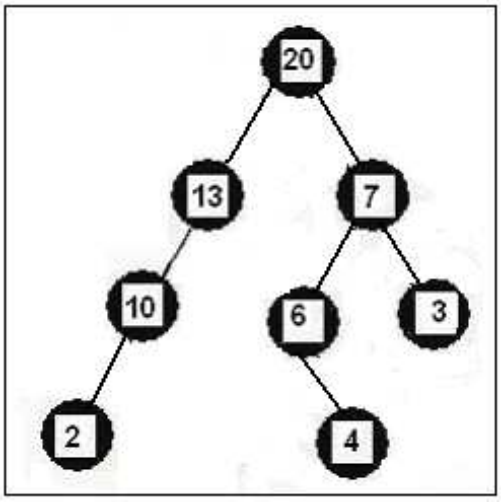
```
public class BinaryTree {
        boolean isEmpty();          // מחזירה אמת אם ורק אם העץ ריק
        BTN get_root();             // מחזירה מצביע לצומת השורש
        BinaryTree left();          // מחזירה את תת העץ השמאלי
        BinaryTree right();         // מחזירה את תת העץ הימני
        int size();                 // מחזירה את מספר הצמתים בעץ
}
```

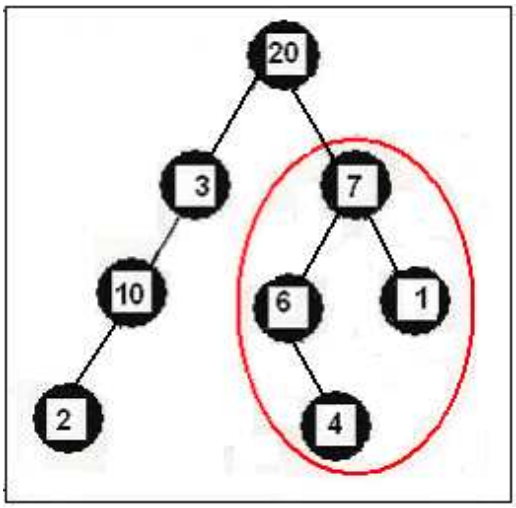
למחלקה צומת של עץ בינארי השיטות הבאות:

```
public class BTN {
        BTN left_node();            // מחזירה את צומת הבן השמאלי
        BTN right_node();           // מחזירה את צומת הבן הימני
        int data();                 // מחזירה את הערך של בצומת
}
```

הגדרה: עץ בינארי הינו *ערימה* אם הערך בכל צומת גדול מכל שאר האיברים בתת העץ המושרש בו (שהוא שורשו).

דוגמא:





העץ הימני הינו ערימה.

העץ שמאלי אינו ערימה שכן תת העץ שבשרשו הערך 3 מכיל ערך הגדול ממנו – 10 (לעומת זאת העץ המוקף בעיגול הוא ערימה (בת 4 איברים).

<u>סעיף א' (11 נקודות)</u>

במחלקה Binary Tree, כתוב שיטה boolean isHeap() המחזירה אמת אם"ם העץ (this) הינו ערימה.

<u>סעיף ב' (20 נקודות)</u>

במחלקה Binary Tree, כתוב שיטה int maxHeapSize() המחזירה את מספר האיברים בתת העץ הגדול ביותר של עצם המפתח המהווה ערימה.

למשל: בדוגמא לעיל, עבור העץ הימני השיטה תחזיר 8, ועבור העץ השמאלי תחזיר 4.

להזכירכם: ניתן להשתמש בשיטה הסטטית Math.max(int a, int b), המקבלת שני שלמים ומחזירה את הערך הגדול ביניהם.

תשובה 4

נוסח ראשון:

```
public boolean isHeap() {
	if(isEmpty()) return true;
	boolean ans = true;
	if(!left().isEmpty())
		ans = left().isHeap() & root() > left().root();
	if(!right().isEmpty())
		ans &= right().isHeap() & root() > right().root();
	return ans;
}

public int maxHeapSize() {
	if(isEmpty()) return 0;
	if(isHeap()) return size();
	return Math.max(left().maxHeapSize(), right().maxHeapSize());
}
```

נוסח שני:

```
public boolean isHeap() {
	if(isEmpty()) return true;
	return isHeap(_root);  }

private boolean isHeap(BTN curr) {
	boolean ans = true;
	BTN ls = curr.left_node(), rs = curr.right_node();
	if(ls!=null) {ans = isHeap(ls) && curr.data() > ls.data();}
	if(rs!=null) {ans &= isHeap(rs) && curr.data() > rs.data();}
	return ans;
}
public int maxHeapSize() {
	if(isEmpty()) return 0; return mhs(_root); }
public int mhs(BTN c) {
	if(c==null) return 0;
	if(c.isLeaf()) return 1;
	else if(isHeap(c)) return size(c);
		else return Math.max(mhs(c.left_node()),mhs(c.right_node()));
}
```

מבוא למדעי המחשב 202-1-101-1

מבחן מועד ב' - סמסטר א' תשס"ד – דף תשובות

חדר: _____ **:מספר נבחן :** _____

ציון	4ב	4א	3	2ב	2א	1

זכרו: מספר השורות העומדות לרשותכם רומז על אורך הקוד הנדרש.

אין צורך להעתיק את חלקי הקוד המופיעים בשאלה. **רק דף התשובות יבדק.** מחברות הטיוטא יגרסו.

תשובה 1:	**תשובה 3:**
תשובה 2 סעיף א:	**תשובה 2 סעיף א המשך:**
	תשובה 2 סעיף ב:
תשובה 4 סעיף א:	**תשובה 4 סעיף ב:**