מבוא למדעי המחשב
202-1-1011

תאריך הבחינה: 16.3.2011

שמות המרצים: מר שי זקוב
ד"ר פז כרמי
פרופ' מייק קודיש
ד"ר חן קיסר
ד"ר צחי רוזן

שם הקורס: מבוא למדעי המחשב

מספר הקורס: 202-1-1011

שנה: 2011 סמסטר: א' מועד: ג'

משך הבחינה: **3 שעות**

חומר עזר: **אסור**

אנא קראו את ההוראות שלהלן בעיון:

- במבחן זה 6 שאלות. ענו על כולן.
- בשאלות **שבהן לא מצוין אחרת**, הנכם יכולים לבחור בפתרון רקורסיבי או בפתרון לא רקורסיבי.
- רשמו את תשובותיכם על גבי טופס הבחינה בלבד. המחברת שקיבלתם היא מחברת טיוטה, והיא לא תימסר כלל לבדיקה.
- החשוב ביותר הוא נכונות הקוד. עם זאת, יעילות, סגנון, וכתיבה ברורה חשובים גם הם, ולכן **תשובה לא יעילה או לא ברורה** עלולה לגרום להפחתת נקודות.
- בשאלות התכנות המקום המוקצה בשאלון לתשובה **מספיק** עבור תשובה המנוסחת כראוי.
- הקפידו על כתב יד ברור. **תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא**. כתבו פתרון לשאלה קודם כול במחברת הטיוטה, ורק לאחר מכן העתיקו פתרון נקי לדף התשובות.
- **אין להוסיף פונקציות עזר** אלא אם כן הדבר נתבקש באופן מפורש בשאלה.
- אין לרשום הערות אך יש לדאוג שהקוד יהיה ברור לחלוטין ללא הערות (ריווח, שמות משתנים, וכד').
- אנא קראו שאלה עד תומה בטרם תענו עליה.
- אם אינכם יודעים את התשובה לשאלה כלשהי, רשמו "לא יודע/ת" (במקום תשובה) ותזכו ב- 20% מניקוד השאלה.

בהצלחה!

<u>שאלה 1 (15 נקודות)</u>

תהי נתונה מחרוזת תווים s. נגדיר: **החלפת ראי** של s היא מחרוזת המתקבלת מ- s ע"י החלפה של שני תווים שנמצאים במרחק שווה מהאמצע של s. למשל "ba" היא החלפת ראי של "ab" ו- "cba" היא החלפת ראי של "abc".

נגדיר: **שתי מחרוזות שוות עד כדי החלפת ראי** אם ניתן להשוות אותן באמצעות 0 או יותר החלפות ראי.

<u>השלימו</u> **<u>בצורה לא רקורסיבית</u>** את הפונקציה eq הבאה המקבלת כפרמטרים שתי מחרוזות s1 ו-s2 ומחזירה true אם"ם s1 ו- s2 שוות עד כדי החלפות ראי.

למשל:

```
eq("", "") → true (החלפות 0)
eq("a", "a") → true (החלפות 0)
eq("ab", "ba") → true (החלפה אחת)
eq("abc", "abc") → true (החלפות 0)
eq("cba", "abc") → true (החלפה אחת)
eq("cbefda", "adefbc") → true (החלפות 2)
eq("cab", "abc") → false
eq("cbad", "abc") → false
```

הניחו ש- s1 ו-s2 שונים מ- null.

ראו תזכורות למספר שיטות של המחלקה String בדף הבא.

```
private static boolean eq(String s1, String s2) {
  boolean ans = true;
  if (s1.length() != s2.length())
    ans = false;
  else {
    for (int i = 0; ans && i < (s1.length() + 1) / 2; ++i)
      if ((s1.charAt(i) != s2.charAt(i) ||
           s1.charAt(s1.length() - 1 - i) != s2.charAt(s2.length() - 1 - i)) &&
          (s1.charAt(i) != s2.charAt(s2.length() - 1 - i) ||
           s1.charAt(s1.length() - 1 - i) != s2.charAt(i)))
        ans = false;
  }
  return ans;
}
```

`public int` **`length`**`()`

Returns the length of this string.

Returns:
the length of the sequence of characters represented by this object.

`public char` **`charAt`**`(int index)`

Returns the `char` value at the specified index. An index ranges from `0` to `length() - 1`. The first `char` value of the sequence is at index `0`, the next at index `1`, and so on, as for array indexing.

Parameters:
`index` - the index of the `char` value.

Returns:
the `char` value at the specified index of this string. The first `char` value is at index `0`.

Throws:
IndexOutOfBoundsException - if the `index` argument is negative or not less than the length of this string.

`public` String **`substring`**`(int beginIndex, int endIndex)`

Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:
`beginIndex` - the beginning index, inclusive.
`endIndex` - the ending index, exclusive.

Returns:
the specified substring.

Throws:
`IndexOutOfBoundsException` - if the `beginIndex` is negative, or `endIndex` is larger than the length of this `String` object, or `beginIndex` is larger than `endIndex`.

שאלה 2 (10 נקודות)

נתון הקוד הבא:

```
static void sort(int[] arr) {
      if (arr != null) {
            int hi = arr.length - 1;
            int i = hi;
            while (i > 0) {
                  bubble(arr, i);
                  i = i - 1;
            }
      }
}

static void bubble(int[] arr, int to) {
      for (int i = 0; i < to; i = i + 1)
            if (arr[i] > arr[i + 1])
                  swap(arr, i, i + 1);
}

static void swap(int[] arr, int u, int v) {
      int t = arr[u];
      arr[u] = arr[v];
      arr[v] = t;
}
```

בבחינה יש להוסיף הערה שיש להניח ש- arr שונה מ- null ושכל המספרים שונים זה מזה ושהטענה הנשמרת צריכה להיות נכונה לכל קלט אפשרי.

צריך לגבי כל אחת מהטענות הבאות האם היא נשמרת בלולאת ה-while שבפונקציה sort (לפני הפעם הראשונה, ואחרי כל איטרציה).

שימו לב: תשובה נכונה מזכה בנקודה ותשובה לא נכונה מפחיתה ½ נקודה. תשובה לא מסומנת אינה מוסיפה ואינה גורעת. סה"כ הנקודות בשאלה כולה אינו יכול להיות שלילי.

שימו לב: בשאלה זו אין לרשום "לא יודע" על סעיף בודד - רק על השאלה כולה.

א. `hi > 0` — נשמרת / לא נשמרת

ב. `i < hi` — נשמרת / לא נשמרת

ג. `arr.length > i` — נשמרת / לא נשמרת

ד. `arr[0] ≤ arr[1] ≤ … ≤ arr[hi]` — נשמרת / לא נשמרת

ה. `arr[0] < arr[1] < … < arr[hi]` — נשמרת / לא נשמרת

ו. `arr[0] ≤ arr[hi]` — נשמרת / לא נשמרת

ז. `arr[0] ≥ arr[1] ≥ … ≥ arr[i]` — נשמרת / לא נשמרת

ח. `arr[0] ≤ arr[1] ≤ … ≤ arr[i]` — נשמרת / לא נשמרת

ט. `arr[i] ≤ arr[i+1] ≤ … ≤ arr[hi]` — נשמרת / לא נשמרת

י. `arr[i+1] < arr[i+2] < … < arr[hi]` — נשמרת / לא נשמרת

שאלה 3 (20 נקודות)

תהי נתונה קבוצה S של מספרים טבעיים (שלמים גדולים מ- 0), ויהי n מספר טבעי כלשהו. נאמר ש- n **סכום מתוך** S אם ניתן לבטא את n כסכום של מספרים (עם או בלי חזרות) מתוך S.

דוגמה: תהי $S = \{4, 5\}$, אזי למשל 13 הוא סכום מתוך S, שכן $13 = 5 + 4 + 4$, אבל 6 אינו סכום מתוך S, שכן לא ניתן לבטא את 6 כסכום של מספרים מתוך S.

שימו לב: 0 הוא סכום מתוך כל קבוצה S, שכן ניתן לבטא את 0 כסכום של קבוצה ריקה של מספרים. כמו כן לכל טבעי $n \in S$, n סכום מתוך S, שכן ניתן לבטא את n כסכום של עצמו בלבד.

הפונקציה `isSumOf(int n, int [] S)` שלהלן מקבלת כפרמטרים מספר טבעי n וקבוצה S של טבעיים ומחזירה `true` אם"ם n הוא סכום מתוך S.

<u>השלימו בצורה רקורסיבית</u> את הקוד החסר.

הניחו שהפרמטרים תקינים: n טבעי, $S \neq null$ ומכיל רק מספרים טבעיים.

```
private static boolean f(int n, int[] S) {
      return isSumOf(n, S, 0);
}

private static boolean isSumOf(int n, int[] S, int i) {
      if (n == 0)
            return true;
      if (n < 0 || i >= S.length)
            return false;
      return isSumOf(n,         S, i + 1) ||
             isSumOf(n - S[i], S, i);
}
```

שאלה 4 (25 נקודות)

אנא קראו את השאלה עד תומה בטרם תענו עליה.

יהי נתון הממשק Task שלהלן:

```
interface Task {

	String title();
	double cost();
	String report();
}
```

הממשק מתאר משימה:

- title() מחזיר את כותרת המשימה, למשל "הכנת כוס תה".
- cost() מחזיר את עלות המשימה, למשל 20 אגורות.
- report() מדווח על נתוני המשימה.

  הדיווח כולל טקסט מוקף בסוגריים עם הנתונים הבאים בסדר הנתון:
  - כותרת המשימה.
  - עלות המשימה.
  - פירוט נוסף בהתאם לסוג האובייקט באמצעות השיטה description (ראו בהמשך).

השלימו את שלושת המחלקות הבאות כמפורט.

1. מחלקה Mission.
   - מחלקה מופשטת שעומדת בראש ההיררכיה של שלושת המחלקות.
   - המחלקה מממשת את הממשק Task.
   - המחלקה מגדירה בונה ציבורי שמקבל את כותרת המשימה (title).
   - המחלקה מגדירה שיטה אבסטרקטית מוגנת בשם description המאפשרת למחלקות היורשות להוסיף תיאור לדיווח באמצעות report.
2. מחלקה SimpleMission.
   - מחלקה קונקרטית המגדירה משימה פשוטה, כלומר משימה שאינה מורכבת ממשימות משנה.
   - המחלקה מרחיבה את המחלקה Mission.
   - המחלקה מגדירה בונה ציבורי המקבל את כותרת המשימה (title).
   - המחלקה מגדירה שיטה ציבורית void setCost(double cost) המקבלת את עלות המשימה cost.
   - המחלקה אינה מוסיפה פירוט נוסף לדיווח באמצעות הפונקציה report().
3. מחלקה CompoundMission.
   - מחלקה קונקרטית המגדירה משימה מורכבת, כלומר משימה שמורכבת מסדרה של משימות משנה.
   - המחלקה מרחיבה את המחלקה Mission.
   - המחלקה מגדירה בונה ציבורי המקבל את כותרת המשימה (title) ומערך מסוג Task המכיל את סדרת משימות המשנה (subtasks) שמרכיבים את המשימה.
   - העלות (cost) של המשימה היא סכום העלויות של משימות המשנה שלה.
   - המחלקה מוסיפה פירוט נוסף לדיווח באמצעות הפונקציה report() המורכב משרשור (concatenation) של הדיווחים של כל משימות המשנה כסדרן.

שימו לב: בתשובתכם עליכם להשתמש בהורשה ובעיקרון ה- polymorphism בצורה מתאימה כדי למחזר קוד בצורה מקסימאלית.

שימו לב: אין להוסיף לממשק הציבורי של המחלקות שתשלימו דבר מעבר לנדרש במפורש, ועל כל השדות שתגדירו בכל המחלקות להיות שדות פרטיים.

להלן תכנית דוגמה והפלט שלה (ניתן כהערה אחרי כל פקודת הדפסה).

שימו לב: הפלט שלכם אינו צריך להכיל שורות חדשות או הזחות (indentations) כדוגמת הפלט שבדוגמה, אך כנאמר עליו כן להכיל סוגריים.

```
public static void main(String[] args) {

	Task sm1 = new SimpleMission("Boil water");
	((SimpleMission) sm1).setCost(10);
	System.out.println(sm1.report());

	// (Boil water, 10.0,)

	Task sm2 = new SimpleMission("Get a cup");
	((SimpleMission) sm2).setCost(1);
	Task sm3 = new SimpleMission("Measure a tea spoon");
	((SimpleMission) sm3).setCost(1);
	Task sm4 = new SimpleMission("Pour boiled water");
	((SimpleMission) sm4).setCost(10);
	Task[] tasks1 = { sm1, sm2, sm3, sm4 };
	Task cm1 = new CompoundMission("Prepare a cup of tea", tasks1);
	System.out.println(cm1.report());

	// (Prepare a cup of tea, 22.0,
	//    (Boil water, 10.0,)
	//    (Get a cup, 1.0,)
	//    (Measure a tea spoon, 1.0,)
	//    (Pour boiled water, 10.0,))

	Task sm5 = new SimpleMission("Bake a cake");
	((SimpleMission) sm5).setCost(30);
	Task sm6 = new SimpleMission("Serve");
	((SimpleMission) sm6).setCost(5);
	Task[] tasks2 = { sm5, cm1, sm6 };
	Task cm2 = new CompoundMission("Serve A cup of tea with a cake", tasks2);
	System.out.println(cm2.report());

	// (Serve A cup of tea with a cake, 57.0,
	//    (Bake a cake, 30.0,)
	//    (Prepare a cup of tea, 22.0,
	//          (Boil water, 10.0,)
	//          (Get a cup, 1.0,)
	//          (Measure a tea spoon, 1.0,)
	//          (Pour boiled water, 10.0,))
	//    (Serve, 5.0,))


	((SimpleMission) sm6).setCost(15);
	System.out.println(cm2.report());

	// (Serve A cup of tea with a cake, 67.0,
	//    (Bake a cake, 30.0,)
	//    (Prepare a cup of tea, 22.0,
	//          (Boil water, 10.0,)
	//          (Get a cup, 1.0,)
	//          (Measure a tea spoon, 1.0,)
	//          (Pour boiled water, 10.0,))
	//    (Serve, 15.0,))
}
```

```
interface Task {

	String title();
	double cost();
	String report();
}

abstract class Mission implements Task {

	private String title;

	public Mission(String title) {
		this.title = title;
	}

	public String title() {
		return title;
	}

	public String report() {
		return "(" + title + ", " + cost() + "," + description() + ")";
	}

	protected abstract String description();
}

class SimpleMission extends Mission {

	private double cost;

	public SimpleMission(String title) {
		super(title);
	}

	public void setCost(double cost) {
		this.cost = cost;
	}

	public double cost() {
		return cost;
	}

	public String description() {
		return "";
	}
}

class CompoundMission extends Mission {

	private Task[] subtasks;

	public CompoundMission(String title, Task[] subtasks) {
		super(title);
		this.subtasks = subtasks;
	}

	public String description() {
		String s = "";
		for (int i = 0; i < subtasks.length; ++i)
			s += subtasks[i].report();
		return s;
	}

	public double cost() {
		double cost = 0;
```

```
            for (int i = 0; i < subtasks.length; ++i)
                  cost += subtasks[i].cost();
            return cost;
      }
}
```

שאלה 5 (15 נקודות)

תהי נתונה המחלקה LinkedList. השלימו במקומות החסרים את השיטה maxSeqLen המחזירה את האורך של תת-הסדרה הלא יורדת הארוכה ביותר המוכלת ברשימה מסוג LinkedList.

למשל אם הרשימת היא 2→2→6→5→5→3→4→1→ על השיטה להחזיר 4, שכן תת-הסדרה הלא יורדת הארוכה ביותר שמוכלת ברשימה היא 3, 5, 5, 6, ואורכה הוא 4.

אם הרשימת ריקה, על השיטה להחזיר 0.

```
class LinkedList {

	private Link first;

	private static class Link {
		int data;
		Link next;
	}

	public int maxSeqLen() {
		Link p = first;
		int max = 0;
		while (p != null) {
			int prev = p.data;
			int n = 1;
			Link q = p.next;
			while (q != null && q.data > prev) {
				++n;
				prev = q.data;
				q = q.next;
			}
			if (n > max)
				max = n;
			p = q;
		}
		return max;
	}
}
```

.

שאלה 6 (15 נקודות)

עץ חיפוש בינארי של מספרים טבעיים (שלמים גדולים מ- 0) יכול להיות מיוצג ע״י מערך של שלמים מאותחל באפסים (ראו את השיטה isEmpty ו- insert שבהמשך).

להלן מחלקה בשם BST המשתמשת במערך כדי לייצג עץ חיפוש בינארי.

- הבנאי מקבל את גודל (קיבולת) המערך.
- השיטה isEmpty מחזירה תשובה לשאלה האם העץ ריק.
- השיטה הציבורית insert מקבלת כפרמטר מפתח k ומוסיפה את k לעץ.
- השיטה הפרטית insert מקבלת מפתח k ואינדקס i במערך arr ומכניסה את k לתת-העץ המושרש ב- arr[i].
- השיטה isaLeaf מקבלת כפרמטר אינדקס i במערך arr ומחזירה true אם״ם arr[i] הוא עלה בעץ.
- השיטה הציבורית search מקבלת מפתח k ומחזירה true אם״ם k מופיע בעץ.
- השיטה הפרטית search מקבלת מפתח k ואינדקס i במערך arr ומחפשת בצורה היעילה ביותר את k בתת-העץ המושרש ב- arr[i].

ענו על הסעיפים הבאים:

א. (5 נקודות) השלימו את השיטה isaLeaf. **הניחו** שהפרמטר i אינו חורג מתחום המערך.

ב. (10 נקודות) השלימו את השיטה הפרטית search. **הניחו** שהפרמטר i אינו חורג מתחום המערך וש- arr[i] הוא אכן קדקוד בעץ. **שימו לב**: נקודות יינתנו רק על חיפוש יעיל ביותר! סריקה סדרתית של תאי המערך לא תזכה בנקודות.

```
class BST {

	private int[] arr;

	public BST(int capacity) {
		arr = new int[capacity];
	}

	public boolean isEmpty() {
		return arr[0] == 0;
	}

	public void insert(int k) {
		insert(k, 0);
	}

	public boolean search(int k) {
		return search(k, 0);
	}

	private void insert(int k, int i) {
		if (i < arr.length)
			if (arr[i] == 0)
				arr[i] = k;
			else if (arr[i] != k)
				if (k < arr[i])
					insert(k, i * 2 + 1);
				else
					insert(k, i * 2 + 2);
	}

	public boolean search(int k, int i) {
		boolean ans;
```

```
            if (i >= arr.length || arr[i] == -1)
                  ans = false;
            else if (k == arr[i])
                  ans = true;
            else if (k < arr[i])
                  ans = search(k, i * 2 + 1);
            else
                  ans = search(k, i * 2 + 2);
            return ans;
      }

      public boolean isAleaf(int i) {
            boolean ans;
            if (arr[i] == 0)
                  ans = false;
            else
                  ans = (2 * i + 1 >= arr.length || arr[2 * i + 1] == 0) &&
                        (2 * i + 2 >= arr.length || arr[2 * i + 2] == 0);
            return ans;
      }
}
```