תאריך הבחינה: 23.12.2011

שמות המרצים: ד"ר טל גרינשפון
פרופ' משה זיפר
ד"ר פז כרמי
מר עדי סוויסה
פרופ' מייק קודיש
ד"ר חן קיסר
ד"ר צחי רוזן

שם הקורס: מבוא למדעי המחשב

מבוא לתכנות למערכות מידע

מספר הקורס: 202-1-1011, 202-1-1041

שנה: 2012 סמסטר: א' מועד: בוחן

משך הבחינה: **3 שעות**

חומר עזר: **אסור**

<u>בוחן אמצע</u>

אנא קיראו היטב את ההוראות שלהלן:

- במבחן זה 5 שאלות. ענו על כל השאלות. בשאלות בהן לא מצוין אחרת, ניתן לבחור בפתרון רקורסיבי או פתרון שאינו רקורסיבי, לבחירתכם.
- רשמו את תשובותיכם **בדפי התשובות בלבד**. המחברת שקיבלתם היא מחברת טיוטה והיא לא תימסר כלל לבדיקה. בסיום הבחינה נשמור אך ורק את דף התשובות. כל שאר החומר יועבר לגריסה.
- **שימו לב:** החשיבות העליונה היא **לנכונות** הקוד. מאידך, **<u>יעילות</u>, סגנון וכתיבה ברורה** חשובים גם הם, ולכן תשובה יעילה ומסוגננת תזכה בציון גבוה יותר.
- <u>בשאלות התכנות</u>, מספר השורות העומדות לרשותכם בדף התשובות רומז על אורך הקוד הנדרש. הקפידו על כתב יד ברור. **תשובות מסורבלות או ארוכות מדי לא יזכו בניקוד מלא**. כתבו פתרון לשאלה קודם במחברת הטיוטה ורק לאחר מכן העתיקו פתרון נקי לדף התשובות.
- אין להוסיף פונקציות עזר אלא אם נאמר במפורש שמותר.
- אם יש סעיף בשאלה המסתמך על סעיף אחר, מותר להשתמש בו גם אם לא פתרתם את הסעיף האחר.
- **הקפידו לרשום בארבעת דפי התשובות גם את מספר הנבחן ומספר החדר שבו אתם נבחנים.**
- **במידה ואינכם יודעים את התשובה לסעיף כלשהו, רשמו "לא יודע/ת" (במקום תשובה) ותזכו ב- 20% מניקוד הסעיף (מעוגל מטה). אם רשום "לא יודע/ת", ההתייחסות היא לכל הסעיף.**

<u>שאלה 1 (20 נקודות)</u>

א. **(5 נקודות)** השלימו בדף התשובות את הפונקציה **isPrime**(**int** num) המקבלת מספר שלם, num, num>1, ומחזירה ערך בוליאני המציין האם num ראשוני או לא. **<u>אסור</u>** להשתמש במערכים. מותר להשתמש במשתני עזר בתנאי שהם מטיפוס פרימיטיבי. אין לחרוג מן המקום המוקצה בדף התשובות.

דוגמאות:

isPrime(7) => true
isPrime(12) => false

ב. **(15 נקודות)** השלימו את הפונקציה **largestPrimesAverage**(**int** num, **int** k) המקבלת מספר שלם num, num>1, ומספר שלם k, k≥1, ומחזירה את הממוצע של k המספרים הראשוניים הגדולים ביותר שקטנים ממש מ- num. במידה וקיימים פחות מ- k מספרים ראשוניים הקטנים ממש מ- num, על הפונקציה להחזיר את הממוצע שלהם. **<u>אסור</u>** להשתמש במערכים. מותר להשתמש במשתני עזר בתנאי שהם מטיפוס פרימיטיבי. אין לחרוג מן המקום המוקצה בדף התשובות. (זכרו כי 1 איננו מספר ראשוני.) מותר להשתמש בפונקציה מסעיף א'.

דוגמאות:

largestPrimesAverage(8, 3) => 5.0 // (7+5+3)/3 = 5.0
largestPrimesAverage(8, 1) => 7.0 // 7/1 = 7.0
largestPrimesAverage(7, 3) => 3.3333… // (5+3+2)/3 = 3.33…
largestPrimesAverage(20, 5) => 13.4 // (19+17+13+11+7)/5 = 13.4
largestPrimesAverage(12, 7) => 5.6 // (11+7+5+3+2)/5 = 5.6

(שימו לב שבדוגמא האחרונה יש פחות מ-k מספרים ראשוניים)

שאלה 2 (20 נקודות)

השלימו בדף התשובות את הפונקציה **equalPartition**(**int[]** arr) המקבלת מערך של מספרים שלמים, ומחזירה true אם ורק אם ישנו אינדקס i במערך כך שסכום המספרים בתאים מהתא הראשון עד i (לא כולל) שווה לסכום המספרים בתאים מהתא i עד סוף המערך.

ניתן להניח כי המערך המתקבל איננו null.

דוגמאות:

```
int[] arr1 = { 1, 2, 3, 4 };          // false יחזיר equalPartition(arr1)
int[] arr2 = { 1, 4, 2, 3 };          // (1+4=2+3 כי) true יחזיר equalPartition(arr2)
int[] arr3 = { 10, 2, 1, 3, 3, 1 }; // (10=2+1+3+3+1 כי) true יחזיר equalPartition(arr3)
int[] arr4 = { 10, 2, 3, 4 };         // false יחזיר equalPartition(arr4)
int[] arr5 = { 1, 7, 4, 1, 11 };      // (1+7+4=1+11 כי) true יחזיר equalPartition(arr5)
int[] arr6 = { 0, -7, -3, 10 };       // (0=-7+-3+10 כי) true יחזיר equalPartition(arr6)
int[] arr7 = { };                     // (0=0 כי באופן ריק מתקיים) true יחזיר equalPartition(arr7)
int[] arr8 = { 0 };                   // (0=0 כי באופן ריק מתקיים) true יחזיר equalPartition(arr8)
int[] arr9 = { 1, -1 };       // (0=1 + -1 כי באופן ריק מתקיים) true יחזיר equalPartition(arr9)
```

שאלה 3 (20 נקודות)

כתבו פונקציה **static boolean** increment(**int[]** vec, **int** base) המקבלת מערך vec אשר מייצג מספר בבסיס base, ומגדילה באחד את ערך המספר המיוצג. נניח כי הספרה המשמעותית ביותר (most significant digit) הינה במקום ה-0 והספרה המשמעותית פחות (least significant digit) הינה במקום האחרון במערך.

הפונקציה תחזיר true אם ניתן להגדיל את הערך המיוצג באחד, ותעדכן את הערך במערך vec בהתאם. אחרת, היא תחזיר false ותאפס את המערך vec.

<u>ניתן להניח</u> כי base≥2 , המערך vec אינו null והוא מכיל מספר חוקי בבסיס.

<u>דוגמאות</u>:

```
vec={ 0, 2, 9, 9, 9 } , base=10  =>   return true, vec={ 0, 3, 0, 0, 0 }
vec={ 1, 0, 1, 3, 3 } , base=4   =>   return true, vec={ 1, 0, 2, 0, 0 }
vec={ 3, 3, 3, 3, 3 } , base=4   =>   return false, vec={ 0, 0, 0, 0, 0 }
vec={ 0, 1, 2, 1, 2 } , base=4   =>   return true, vec={ 0, 1, 2, 1, 3 }
```

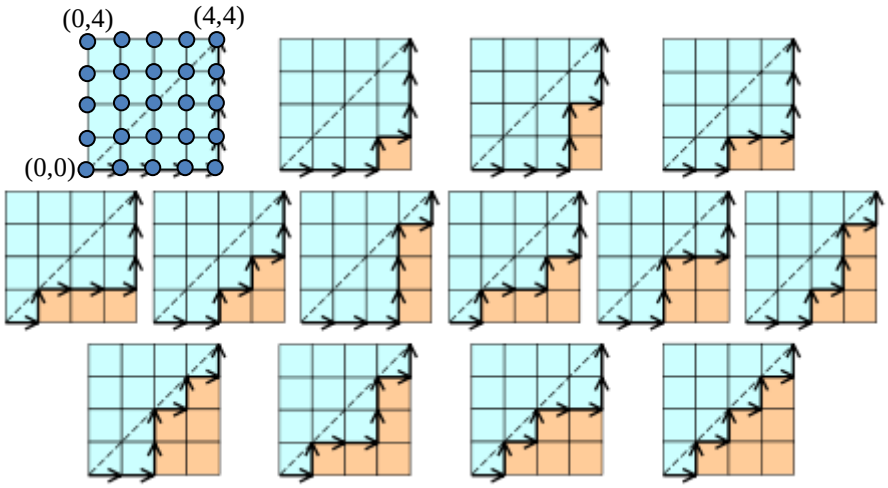
<u>שאלה 4 (20 נקודות)</u>

נתון שריג (grid) בגודל nXn תאים, ובו (n+1)X(n+1) נקודות מפגש (ראה ציור).

נגדיר — מסלול מונוטוני הינו מסלול אשר:

1. מתחיל בפינה השמאלית התחתונה, אשר נסמנה בקואורדינטות (0,0), ומסתיים בפינה הימנית העליונה (n,n),
2. אינו עובר בנקודות הנמצאות מעל לאלכסון אלא רק בנקודות הנמצאות מתחת או על האלכסון,
3. ובנוי אך ורק מצעדים הנעים ימינה או למעלה.

דוגמא למסלולים מונוטוניים עבור n=4:



א. (15 נקודות) השלם את הפונקציות **path**(**int** n) ו-**pathRec**(**int** n, **int** x, **int** y) כך ש-**path** תחזיר את מספר המסלולים המונוטוניים עבור n כלשהו. אין להשתמש בפונקציות נוספות מלבד השתיים הנתונות.

ב. (5 נקודות) האם ניתן לשפר את יעילות הזמן של הפונקציה **pathRec**? אם כן תאר במשפט קצר אחד (ללא קוד) כיצד ניתן לעשות זאת, אם לאו, הסבר במשפט קצר מדוע.

<u>שאלה 5 (20 נקודות)</u>

נתונות הפונקציות insertionSort ו-insert הממשות מיון הכנסה. כמו כן נתונות 9 טענות.

א. (10 נקודות) סמן את כל הטענות שהינן נכונות בכל פעם לפני בדיקת התנאי בלולאת ה-while בפונקציה insert (ה-while המסומן בחץ) כאשר מניחים שהפונקציה insert נקראת מתוך insertionSort.

ב. (10 נקודות) סמן את כל הטענות שהינן נכונות בכל פעם לפני בדיקת התנאי בלולאת ה-while בפונקציה insert (ה-while המסומן בחץ) כאשר הפונקציה insert **אינה נקראת** בהכרח מתוך insertionSort.

הטענות (i ו-j הם אילו שבפונקציה insert):

1) value גדול מכל אברי המערך בתחום 0 עד i כולל
2) value גדול מכל אברי המערך בתחום 0 עד i לא כולל i
3) value גדול מכל אברי המערך בתחום i+1 עד גודל המערך פחות אחד (כולל)
4) value קטן מכל אברי המערך בתחום 0 עד i כולל
5) value קטן מכל אברי המערך בתחום i+1 עד j כולל
6) value קטן מכל אברי המערך בתחום i+1 עד גודל המערך פחות אחד (כולל)
7) value נמצא במקומו במערך הממוין (כולו)
8) המערך ממוין
9) המערך ממוין בתחום 0 עד i לא כולל i

```
public static void insertionSort(int [] arr){
        for (int i = 1; i<arr.length; i = i+1)
                insert(arr, i);
}

public static void insert(int[] arr, int j) {
        int value = arr[j];
        int i=j;
        while (i > 0 && arr[i-1] > value) {
                arr[i] = arr[i-1];
                i = i-1;
        }
        arr[i] = value;
}
```

שאלה 5 (20 נקודות) [IPIS]

בהינתן מערך של משקולות שלמים וגדולים מאפס ומשקל יעד sum ≥0, נרצה לבדוק האם ניתן להרכיב מהמשקולות משקל השווה למשקל הנתון.

דוגמא לקלט: weights={1,7,9,3} ו- sum = 12

במקרה זה התשובה היא true כי ניתן לחבר את המשקולות 9 ו 3 ולקבל את הסכום 12. במידה ו-sum=2 עבור אותו המערך, התשובה היא false.

להלן הפתרון של בעיה זו בדיוק כפי שראיתם בתרגול:

```
public static boolean calcWeightsInit(int[] weights, int sum) {
      return calcWeights(weights, 0, sum);
}

public static boolean calcWeights(int[] weights, int i, int sum) {
boolean res = false;
if (sum == 0)
      res = true;
else if (i >= weights.length)
      res = false;
else
      res = (calcWeights(weights, i + 1, sum - weights[i]) ||
             calcWeights(weights, i + 1, sum));
return res;
}
```

בהנחה שגודל מערך weights הינו n וכי sum≥0 ענה על השאלות הבאות:

א. מהו סדר גודל הקריאות ל-`calcWeights` (שימו לב: לא `calcWeightsInit`) במקרה הגרוע ביותר, כלומר המקרה שיביא למספר קריאות מקסימאלי?

ב. מהו המקרה הגרוע ביותר, כלומר המקרה שיביא למספר קריאות מקסימאלי?

ג. מהו סדר גודל הקריאות ל-`calcWeights` במקרה הטוב ביותר, כלומר המקרה שיביא למספר קריאות מינימאלי?

ד. מהו המקרה הטוב ביותר, כלומר המקרה שיביא למספר קריאות מינימאלי?