מבוא למדעי מחשב מ' / ח' (234114 / 234117)

סמסטר חורף תשס"ו

פתרון מבחן מסכם מועד ב', 22 מרץ 2006

מספר סטודנט	*שם משפחה*	*שם פרטי*

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני.

הנחיות והוראות:

- מלאו את הפרטים בראש דף זה.
- בדקו שיש 10 עמודים (5 שאלות) במבחן, כולל עמוד זה.
- כתבו את התשובות על טופס המבחן בלבד, במקומות המיועדים לכך. שימו לב שהמקום המיועד לתשובה אינו מעיד בהכרח על אורך התשובה הנכונה.
- העמודים הזוגיים בבחינה ריקים. ניתן להשתמש בהם כדפי טיוטה וכן לכתיבת תשובותיכם. סמנו טיוטות באופן ברור על מנת שהן לא תיבדקנה.
- יש לכתוב באופן ברור, נקי ומסודר. ניתן בהחלט להשתמש בעיפרון ומחק.
- אין לכתוב הערות והסברים לתשובות אם לא נתבקשתם מפורשות לכך.
- בכל השאלות, הינכם רשאים להגדיר (ולממש) פונקציות עזר כרצונכם.
- **אין להשתמש בפונקציות ספריה או בפונקציות שמומשו בכיתה אלא אם צוין אחרת בשאלה.**

שאלה	ערך	הישג	בודק
1	20		
2	20		
3	20		
4	20		
5	20		
סה"כ	100		

צוות הקורס 234114
מרצים: סאהר אסמיר, פרופ' רון קימל (מרצה אחראי).
מתרגלים: עידן בן-הרוש, גיא סלע, ולנטין קרבצוב, מרק גינזבורג, רן רובינשטיין (מתרגל אחראי).

צוות הקורס 234117
מרצים: ארז חדד, ויטלי סקצ'ק, פרופ' רון קימל (מרצה אחראי).
מתרגלים: שיאון שחורי, רג'א ג'יריס, אסנת טל, מרק גינזבורג, ארקדי פיורו, רן רובינשטיין (מתרגל אחראי).

בהצלחה!

שאלה 1 (20 נקודות)

נתונים שלושה מערכים של מספרים שלמים a b ו-c, ואורכיהם n m ו-r בהתאמה. כל מערך ממוין בסדר עולה-ממש, וללא חזרות. עליכם לחשב מערך פלט d שיכיל את **החיתוך** של שלושת המערכים; כלומר, ערך כלשהו יופיע ב-d אם ורק אם הוא מופיע בכל שלושת מערכי הקלט. על מערך הפלט d להיות ממוין אף הוא בסדר עולה וללא חזרות. כמו כן על הפונקציה להחזיר את מספר האיברים במערך d כערך ההחזרה שלה. לדוגמה, עבור מערכי הקלט:

a:	1	2	6	9			
b:	2	5	6	7	8	9	11
c:	2	6	8	9			

יוחזר 3, ויתקבל מערך הפלט הבא:

d:	2	6	9

דרישות סיבוכיות: זמן O(N) (כש-N הוא סכום אורכי שלושת מערכי הקלט), מקום נוסף O(1).

```
int intersect3(int a[], int n, int b[], int m,
               int c[], int r, int d[]) {

  int i=0, j=0, k=0, pos=0;

  while (i<n && j<m && k<r) {
    if (a[i]==b[j] && b[j]==c[k]) {
      d[pos++]=a[i++];
      j++; k++;
    }
    else if (a[i]<=b[j] && a[i]<=c[k])
      i++;
    else if (b[j]<=a[i] && b[j]<=c[k])
      j++;
    else
      k++;
  }

  return pos;
}
```

פתרון חילופי:

```
int intersect3(int a[], int n, int b[], int m,
               int c[], int r, int d[]) {

  int i=0, j=0, k=0, pos=0;

  while (i<n && j<m && k<r) {
    int minval = min(a[i],b[j],c[k]), count=0;
    if (a[i]==minval) { count++; i++; }
    if (b[j]==minval) { count++; j++; }
    if (c[k]==minval) { count++; k++; }
    if (count==3) d[pos++] = a[i-1];
  }
  return pos;
}

int min(int x, int y, int z) {
  return min2(min2(x,y),z);
}

int min2(int x, int y) {
  return (x<y ? x : y);
}
```

שאלה 2 (20 נקודות)

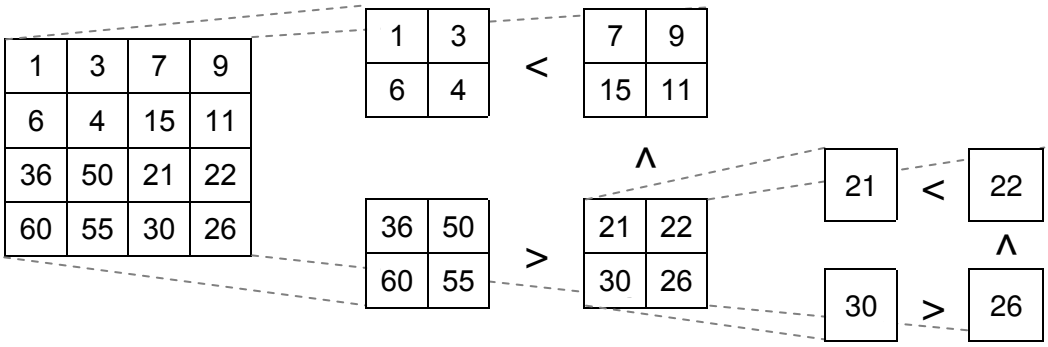
בשאלה זו נתייחס למערכים דו-ממדיים **ריבועיים** בגודל NxN (כאשר N הוא קבוע המוגדר כ-define#). לצורך השאלה, נניח כי N הוא חזקה שלמה של 2. עבור מערך כזה, נגדיר חלוקה פנימית שלו לארבעה רובעים בגודל N/2xN/2, ממוספרים מ-1 עד 4, באופן הבא:

1	2
4	3

נאמר שהמערך הוא **סיבובי** אם כל האיברים ברובע 1 קטנים ממש מכל אלו שברובע 2, אלו שברובע 2 קטנים ממש מכל אלו שברובע 3, ואלו שברובע 3 קטנים ממש מכל אלו שברובע 4. למשל, המערך הבא הוא סיבובי:

1	5
9	7

לשם הנוחות, נגדיר גם כל מערך בגודל 1x1 כמערך סיבובי. כעת, נאמר שמערך NxN הוא **ממוין-סיבובית** אם הוא סיבובי, ארבעת הרובעים שלו סיבוביים, וכן הלאה עד לרובעים בגודל 1x1. לדוגמה, המערך הבא ממוין-סיבובית:



עליכם לממש פונקציה לחיפוש במערך ממוין-סיבובית (בעמוד הבא). הפונקציה מקבלת כפרמטר את המערך הדו-ממדי a[N][N], ואת הערך לחיפוש x. במידה ו-x נמצא במערך, הפונקציה תחזיר 1 ותכתוב את הקואורדינאטות שלו למשתנים i,j הניתנים כפרמטרים. במידה ו-x אינו במערך, הפונקציה תחזיר 0 ואין חשיבות לתוכן של i,j.

דרישות סיבוכיות: על הפונקציה לעבוד בסיבוכיות זמן טובה ככל הניתן. פתרון לא אופטימאלי יזכה בניקוד חלקי לכל היותר. כמו כן על הפונקציה לעבוד בסיבוכיות מקום נוסף O(1).

```
int find2d(int a[N][N], int x, int* i, int* j) {

  int top=0, left=0, size=N;

  while (size > 1) {
    if (x > a[top + size/2 - 1][left]) {
      // not in 1st quarter
      if (x <= a[top + size/2 - 1][left + size/2]) {
        // 2nd quarter
        left += size/2;
      } else if (x <= a[top + size - 1][left + size/2]) {
        // 3rd quarter
        top += size/2;
        left += size/2;
      } else
        // 4th quarter
        top += size/2;
    }
    size /= 2;
  }

  if (a[top][left] == x) {
    *i = top;
    *j = left;
    return 1;
  } else
    return 0;
}
```

שאלה 3 (20 נקודות)

סעיף א (10 נקודות)

נתון **מערך של מחרוזות** []strings באורך N (כש-N מוגדר כ-#define), **ומפתח** key שאף הוא מחרוזת. עליכם לממש פונקציה המחשבת לכל מחרוזת במערך strings כמה תווים משותפים יש לה עם המחרוזת key (כאשר חזרות נספרות, ראו דוגמה בהמשך). על הפונקציה לרשום את התוצאות למערך vals, כש-vals[i] מכיל את מספר התווים המשותפים למפתח עם המחרוזת ה-i. **שימו לב:** אין לשנות את מחרוזות הקלט.

לדוגמה, למחרוזת "Good morning Israel!" ולמפתח "oops" יש 3 תווים משותפים: שני 'o' ו-'s' אחד. לעומת זאת עם המפתח "!ooooops" יש לה 5 תווים משותפים: שלושה 'o', 's' אחד ו-'!' אחד.

דרישות סיבוכיות: סיבוכיות מקום נוסף O(1), וסיבוכיות זמן O(N*m+k) כש-m הוא אורך המחרוזת הארוכה ביותר ו-k הוא אורך המפתח. בשאלה זו ניתן להתייחס למספר התווים האפשריים (256) כקבוע.

```
void keycount(char* strings[N], char* key, int vals[N]) {

  int keycount[256]={0}, i, j;

  for (i=0; key[i]!=0; ++i)
    keycount[key[i]]++;

  for (j=0; j<N; ++j) {
    int strcount[256]={0};
    for (i=0; strings[j][i]!=0; ++i) {
      strcount[strings[j][i]]++;
    }
    vals[j]=0;
    for (i=0; i<256; ++i) {
      vals[j] += min(keycount[i], strcount[i]);
    }
  }
}

int min(int x, int y) {
  return (x<y ? x : y);
}
```

סעיף ב (10 נקודות)

בשאלה זו נתון מערך של מחרוזות ומפתח, כמתואר בסעיף א'. יש לממש פונקציה **שממיינת** את מערך המחרוזות בסדר עולה על פי ערכן של המחרוזות, כאשר ערכה של מחרוזת מוגדר כמספר התווים שמשותפים לה עם המפתח. בסעיף זה ניתן להשתמש בפונקציה מסעיף א' גם אם לא פתרתם אותו.

דרישות סיבוכיות: סיבוכיות מקום נוסף O(N). סיבוכיות זמן $O(N*m+k+N^2)$, כש-N, m ו-k כפי שהוגדרו בסעיף א'.

```
void keysort(char* strings[N], char* key) {

  int vals[N], i, j;

  keycount(strings, key, vals);

  for (i=0; i<N-1; ++i) {
    for (j=0; j<N-i-1; ++j) {
      if (vals[j] > vals[j+1]) {
        swapint(vals+j,  vals+j+1);
        swapstr(strings+j, strings+j+1);
      }
    }
  }
}

void swapint(int* a, int* b) {
  int tmp=*a; *a=*b; *b=tmp;
}

void swapstr(char** a, char** b) {
  char* tmp=*a; *a=*b; *b=tmp;
}
```

שאלה 4 (20 נקודות)

כתבו את סיבוכיות הזמן והמקום הנוסף של הפונקציות f1 ו-f2:

```
void f1(int n)
{
  int i, k=0;

  for (i=0; i<n; i++)
    k += g1(i+1)-g1(i);

  while (k>0) {
    printf("k= %d", k);
    k--;
  }
}

int g1(int n)
{
  int i, j=0;
  for (i=n; i>0; i--)
    j += i;
  return (int)sqrt(2*j-n);
}
```

סיבוכיות זמן: $\Theta($ n^2 $)$ סיבוכיות מקום נוסף: $\Theta($ 1 $)$

```
void f2(int n)
{
  if (n<=1)
    return;
  g2(n, n/3);
}

void g2(int n, int m)
{
  int i=1;
  while (m < n) {
    m += i;
    i++;
  }
  f2(n/2);
}
```

סיבוכיות זמן: $\Theta($ $\sqrt{n}$ $)$ סיבוכיות מקום נוסף: $\Theta($ $\log n$ $)$

שאלה 5 (20 נקודות)

מעוניינים להרכיב צוות שמירה למוזיאון. ברחבי המוזיאון פזורות עמדות שמירה, שבכל אחת ניתן לשים שומר אחד בלבד. הדרישה היא שלכל פסל במוזיאון יהיו לפחות שני שומרים המסוגלים לראות אותו. ברצוננו למצוא את צוות השומרים **המינימאלי** הדרוש על מנת לשמור על המוזיאון. לשם כך, נתונה מטריצה a בגודל $M \times N$, כאשר N מייצג את מספר הפסלים ו-M את מספר עמדות השמירה. a[i][j] מכיל 1 אם נקודת השמירה ה- i משקיפה על הפסל ה- j, ו- 0 אחרת.

השלימו את הפונקציה findGuardTeam למטה, ואת פונקצית העזר בה היא משתמשת (בעמוד הבא), אשר מקבלת כפרמטר את המטריצה a ומחזירה את גודל צוות השמירה המינימלי המקיים את הדרישות. אם לא קיים כזה, הפונקציה מחזירה את הערך המיוחד M+1. כמו כן, במידה וניתן למצוא צוות שמירה העומד בדרישות, על הפונקציה לכתוב את מיקומי השומרים למערך result, שיכיל 1 במקום ה- i אם יש למקם שומר בעמדה ה- i, ו- 0 אחרת.

לנוחותכם, ניתן להשתמש בפונקצית העזר הבאה, המקבלת מערך באורך n של מספרים שלמים וערך x, ומחזירה 1 אם כל איברי המערך גדולים או שווים ל-x, ו-0 אחרת.

```
int atleastX(int arr[], int n, int x)
```

הערה: מספר השורות הריקות לא בהכרח מעיד על מספר שורות הקוד שיש לכתוב, ואורך הקו אינו בהכרח מעיד על אורך השורה שיש להשלים. ניתן להניח כי N ו-M הם קבועים המוגדרים כ-#define.

```
int findGuardTeam (int a[M][N], int result[M])
{
   int aux[_____N_____]= {0}, i;

   for ( i = 0 ; i < M ; ++i )
      result[i] = _______0_________;

   return guardTeamAux(a, result, aux, 0);
}
```

```
int guardTeamAux (int A[M][N], int result[M], int aux[], int k)
{
      int min_with_curr, min_without_curr, i;
      int res_with_curr[        M        ] = {0};

      if ( atleastX(aux, N, 2) == 1          ) return 0;
      if ( k == M                            ) return M+1;

      for ( i = 0 ; i < N ; ++i ) {
            aux[i] += A[k][i]                ;
      }

      min_with_curr =
            guardTeamAux ( A, res_with_curr, aux    , k+1   );

      for ( i = 0 ; i < N ; ++i ) {
            aux[i] -= A[k][i]                ;
      }
      min_without_curr =
            guardTeamAux ( A, result        , aux    , k+1   );

      if ( min_with_curr +1 < min_without_curr) {
            ____________________________________
            result[k] = 1;
            for (i=k+1; i<M; ++i) {
               result[i] = res_with_curr[i];
            }
            ____________________________________
            return min_with_curr + 1         ;
      }
      return min_without_curr          ;
}
```