מבוא למדעי המחשב
מועד ב', סמסטר א' תשס"ב, 7/3/02

מרצה: שולי וינטנר.
מתרגלים: שלמה יונה, ליאת לונטל.

משך המבחן: שעתיים וחצי.
חומר עזר: מותר כל חומר עזר, מלבד מחשב.
הנחיות:

1. ודאו כי בטופס שבידיכם 8 עמודים. יש לכתוב את התשובות על גבי טופס המבחן ולהגיש את כל הטופס ואת הטופס בלבד.
2. קראו היטב כל שאלה. ודאו כי אתם מבינים את השאלה לפני שתתחילו לענות עליה.
3. כתבו בכתב יד ברור וקריא. השתמשו בדפי הטיוטה והעתיקו לטופס המבחן רק תשובות סופיות. תשובות לא קריאות לא תיבדקנה.
4. הערות לתשובותיכם ניתן לכתוב בעברית, גם בגוף פונקציות C.
5. אם לא נכתב אחרת, כאשר עליכם להגדיר פונקציה יש להגדיר פונקציה אחת בדיוק. לא ניתן להשתמש בפונקציות חיצוניות.
6. אם לא נכתב אחרת, בתוכניות ניתן להשתמש בפונקציות מתוך הספריות הבאות בלבד:
   a. stdio.h
   b. stlib.h
   c. string.h
   d. ctype.h

בהצלחה!

שאלה	ציון
1	/20
2	/20
3	/30
4	/24
5	/6
סה"כ	/100

שאלה 1- 20 נקודות:

נתון מערך המוגדר כך: `int array[N]`. `N` הוא קבוע המוגדר ב-`#define`.

כתבו פונקציה המקבלת מערך כזה ומחזירה 1 אם ורק אם אברי המערך הם פרמוטציה של המספרים: 0,1,…,N-1. על הפונקציה להחזיר 0 אחרת.

לדוגמה, אם N=5 והמערך הוא :

3	1	4	2	0

הפונקציה תחזיר 1.

אם המערך הוא:

0	1	4	3	3

הפונקציה תחזיר 0.

אם המערך הוא:

0	-1	-2	-3	-4

הפונקציה תחזיר 0.

על הפונקציה לעבוד בזמן **O(N)**. פתרונות בסיבוכיות גבוהה יותר לא יתקבלו.

```
int permutation (int array[N])

{
  int temp[N];
  int i;

  for (i=0; i<N; i++)
    temp[i] = -1;
  for (i=0; i<N; i++)
    if (array[i] >= 0 && array[i] < N)
      temp[array[i]] = array[i];
    else
      return 0;
  for (i=0; i<N; i++)
    if (temp[i] == -1)
      return 0;
  return 1;
}
```

שאלה 2- 20 נקודות:

בשאלה זו ניתן להשתמש בכל הפונקציות שהודגמו בהרצאה, ללא צורך להגדיר אותן. אם הנכם משתמשים בפונקציה חיצונית כזו, הצהירו עליה, הסבירו בהערה מה היא מבצעת וקבעו את סיבוכיותה.

נתונים שני מערכים באורך זהה המוגדרים כך: `int array[N], int barray[N]`.
כתבו פונקציה המקבלת שני מערכים כאלו, ומחזירה 1 אם ורק אם כל איבר ב-array מופיע ב-barray.
על הפונקציה להחזיר 0 אחרת.
לדוגמה: עבור N=4 אם המערכים הם:

array:

1	2	4	2

barray:

1	4	8	2

הפונקציה תחזיר 1.

על הפונקציה לעבוד בזמן **O(N logN)**. פתרונות בסיבוכיות גבוהה יותר לא יתקבלו.

```
/* search 'n' in 'v'; O(log |v|) */
int binsearch (int n, int v[], int low, int high);
/* sort 'array' of length 'n' using merge; O(n log n) */
void mergesort (int array[], int n);

int contained (int array[ ], int barray[ ])
{
  int i;

  mergesort(barray, N);
  for (i=0; i<N; i++) {
    if (binsearch (array[i], barray, 0, N) < 0)
      return 0;
  }
  return 1;
}
```

שאלה 3- 30 נקודות:

נייצג **תמונה בשחור לבן** ע"י מערך דו-ממדי, שכל תא בו מכיל 1 עבור נקודה שחורה, ו-0 עבור נקודה לבנה. נתונה ההגדרה:

```
typedef enum {WHITE, BLACK} Pixel;
typedef Pixel Picture[N][M];
```

כאשר N ו-M מוגדרים ב-define#.

נגדיר את **מטריצת השליטה** של תמונה כמערך דו-ממדי של שלמים, בו במקום ה-[i][j] מוצב הערך n אם ורק אם התא ה-[i][j] של התמונה הוא נקודה שחורה, וגם n-1 התאים שתחתיו הם נקודות שחורות.

לדוגמה, מטריצת השליטה של התמונה p:

0	1	1	0	0
0	1	1	1	0
0	1	1	1	0
0	0	1	0	0

היא d:

0	3	4	0	0
0	2	3	2	0
0	1	2	1	0
0	0	1	0	0

נתונה ההגדרה:

```
typedef int Dominance[N][M];
```

(1 – 10 נקודות:

כתבו פונקציה אשר מקבלת תמונה ובונה את מטריצת השליטה שלה. פתרונות בסיבוכיות גבוהה מ-O(NxM) יזכו בניקוד חלקי.

```
void compute_dominance (Picture p, Dominance d)

{
  int i,j;

  for (j=0; j<M; j++) { /* base case: bottom row */
    d[N-1][j] = p[N-1][j]; /* d is 1 if p is black, 0 otherwise */
  }
  for (i=N-2; i>=0; i--) { /* compute upper rows */
    for (j=0; j<M; j++) { /* for each column */
      d[i][j] = (p[i][j] ? d[i+1][j] + 1 : 0);
    }
  }
}
```

2) – 15 נקודות:

כתבו פונקציה המקבלת תמונה, את מטריצה השליטה שלה וקואורדינטות של תא בתמונה, ומחזירה את שטחו של המלבן השחור הגדול ביותר שהתא הנתון הוא הקודקוד השמאלי העליון שלו. למשל, בהינתן תמונת הדוגמה p ומטריצת השליטה d שלה שבעמוד הקודם, תחזיר הפונקציה ערכים כדלקמן:

```
max_rect_at_point (p, d, 0, 0) → 0
max_rect_at_point (p, d, 0, 1) → 6
max_rect_at_point (p, d, 0, 2) → 4
max_rect_at_point (p, d, 1, 2) → 4
max_rect_at_point (p, d, 2, 2) → 2
max_rect_at_point (p, d, 3, 2) → 1
```

פתרונות בסיבוכיות גבוהה מ-O(M+N) יזכו בניקוד חלקי.

```
int max_rect_at_point (Picture p, Dominance d, int i, int j)
{
  int k, length, width, area;

  if (!p[i][j]) {
    return 0;
  }
  area = length = d[i][j]; /* for a rectangle of width 1,
                              area = length */
  k = j+1; width = 2; /* try rectangles of increasing width */
  while (k < M && p[i][k]) { /* as long as the line is black */
    length = MIN (length, d[i][k]);
    area = MAX (area, width*length);
    k++; width++;
  }
  return area;
}
```

3) – 5 נקודות:

כתבו פונקציה המקבלת תמונה ומחזירה את שטח המלבן השחור הגדול ביותר בתמונה. ניתן להשתמש בכל אחת מן הפונקציות של הסעיפים הקודמים, גם אם לא הגדרתם אותן.

```
int max_rect (Picture p)
{
  int i,j,size;
  int max = 0;
  Dominance d;

  compute_dominance (p,d);
  print_dominance (d);

  for (i=0; i<N; i++) {
    for (j=0; j<M; j++) {
      size = max_rect_at_point (p, d, i, j);
      if (size > max) {
          max = size;
      }
    }
  }
  return max;
}
```

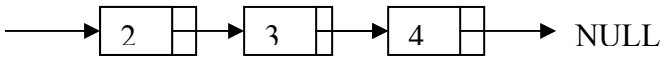
שאלה 4- 24 נקודות:

נתונה רשימה מקושרת אשר כל צומת בה מוגדר כך:

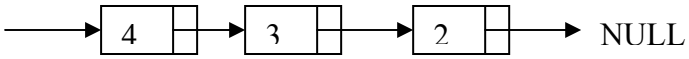
```
typedef struct node {
     int data;
     struct node *next;
} Node;
```

ידוע כי ברשימה לפחות איבר אחד, וכי האיבר האחרון ברשימה מצביע ל-NULL. הפונקציה ה**רקורסיבית** reverse_list מקבלת מצביע לתחילת רשימה מקושרת, והופכת את סדר האברים ברשימה.

לדוגמה: אם הרשימה היא הרשימה הבאה:



לאחר הקריאה לפונקציה הרשימה תראה כך:



שימו לב כי על הפונקציה לשנות את הרשימה עצמה. מותר לשנות את ערכי המצביעים בכל רשומה!
השלימו את הגדרת הפונקציה. בכל משבצת יש לכתוב ביטוי אחד בדיוק.

```
Node reverse_list (Node *list)

{

  Node *p;


  if (____list->next == NULL____________) {

    return (________list_________________);

  } else {

    p = reverse_list (____list->next_____);

    list->next->next = list_______________;

    list->next = NULL_____________________;

    return (____p_________________________);

  }

}
```

שאלה 5- 6 נקודות:

השלימו את הפלט במקומות המתאימים:

```
#include <stdio.h>
int x=10;
int y=11;

int f1(int x, int y) {
     if(x/2==y/2)
          return x/2;
     y/=2;
     return y;
}

int f2(int *x, int y) {
     if ((*x % 10) < (y/2))
          return *x=y;
     return y/2;
}

void f3() {
     int z;
     x=y=z=200;
     return;
}

int main( ) {
     int z=6,i=7,j=8;
     z+=f1(i,j);
     printf("z=%d  i=%d  j=%d\n",z,i,j);
     z=8;
     z=f2(&i,j);
     printf("z=%d  i=%d  j=%d\n",z,i,j);
     printf("x=%d\n",x);

     z=100;
     f3();
     printf("x=%d  y=%d  z=%d\n",x,y,z);


     {
          int x=20;
          int *y=&x;
          x+=5;
          printf("x=%d  *y=%d\n",x,*y);
     }
     return 0;
}
```

z= 10 i= 7 j= 8

z= 4 i= 7 j= 8
x= 10

x= 200 y= 200 z= 100

x= 25 *y= 25