מבוא למדעי המחשב
בחינת מועד א', סמסטר א' תשס"ה, 26.1.2005

מרצה: שולי וינטנר.
מתרגל: עזרא דאיה.

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. כאשר עליכם להגדיר פונקציה יש להגדיר פונקציה אחת בדיוק. לא ניתן להשתמש בפונקציות חיצוניות.
6. ניתן להשתמש בפונקציות מתוך הספריות `stdio.h, stdlib.h` ובפונקציה `strlen` בלבד.

בהצלחה!

שאלה	ציון
1	/15
2	/50
3	/35
סה"כ	/100

<u>שאלה 1- 15 נקודות:</u>

הגדירו פונקציה המקבלת מערך של שלמים ואת אורכו ומחזירה מספר שלם כלשהו ש**אינו** איבר במערך.
סיבוכיות זמן: `O(n)`. סיבוכיות מקום: `O(1)`.

שאלה 2- 50 נקודות:

בהינתן שתי מחרוזות, **חיתוך** שלהן הוא תת-סדרה רציפה של תווים המופיעה בשתי המחרוזות. למשל, למחרוזות "hello" ו-"there" מספר חיתוכים שונים, כולל המחרוזת "h", המחרוזת "e" והמחרוזת "he". למחרוזות "hello" ו-"cat" אין אף חיתוך לא-ריק. חיתוך הוא **מקסימלי** אם לא קיים חיתוך ארוך ממנו. למשל, למחרוזות "hello" ו-"there" רק החיתוך "he" הוא מקסימלי.
בשאלה זו עליכם לכתוב תוכנית המקבלת בשורת הפקודה שתי מחרוזות ומדפיסה את אורך החיתוך המקסימלי שלהן. למשל, אם שם התוכנית הוא intersect, הפעלתה תיראה כך:

```
> intersect hello there
2
> intersect yes no
0
> intersect hello hello
5
```

הסעיפים השונים ידריכו אתכם בפתרון הבעיה. קראו את כל השאלה לפני שתתחילו לענות עליה! ניתן להסתמך בכל סעיף על ההנחה ששאר הסעיפים פתורים כהלכה.

א. הצהירו על פונקציה בשם intersect המקבלת שתי מחרוזות באורך לא ידוע ומחזירה מספר שלם המציין את אורך החיתוך המקסימלי שלהן. אין צורך להגדיר את הפונקציה!

ב. הגדירו פונקצית main המקבלת ארגומנטים משורת הפקודה. על main לבדוק שהועבר לה מספר נכון של ארגומנטים, ואם מספר הארגומנטים שגוי, עליה להדפיס הודעת שגיאה ולהחזיר 1. אם מספר הארגומנטים נכון, עליה לקרוא לפונקציה intersect עם הפרמטרים הראויים ולהדפיס את הערך שפונקציה זו מחזירה.

ג. הגדירו פונקציה בשם matalloc המקבלת שני מספרים שלמים, m,n (המייצגים את אורכי שתי מחרוזות הקלט), ומחזירה מצביע לשטח זיכרון רציף המספיק לאחסון מטריצה דו-ממדית של שלמים, בגודל m*n. טיפוס הערך המוחזר מן הפונקציה הוא (int *), כלומר ניתן לחשוב על הפונקציה כאילו היא מחזירה מערך בגודל m*n של שלמים. על הפונקציה להחזיר NULL במקרה של כישלון.

_____
_____
_____
_____
_____
_____
_____
_____
_____

ד. הפונקציה matalloc מחזירה מצביע לשטח זיכרון רציף המספיק לאחסון m*n שלמים, אולם היא מחזירה את השטח כמערך חד-ממדי, בעוד שברצוננו לחשוב עליו כעל מערך דו-ממדי, שבו m שורות, כל אחת באורך n. בהנחה ש-n קבוע, הגדירו מקרו בשם ind עם שני פרמטרים, i ו-j, המחזיר את האינדקס (היחיד!) במערך חד-ממדי של האיבר ה-[i][j] במערך דו ממדי ששורותיו באורך n.

```
#define ind(i,j) ____________________________________________
```

אם a הוא מערך חד-ממדי שהוקצה על ידי הפונקציה matalloc, כיצד תשתמשו במקרו כדי לגשת לאיבר במקום ה-[2][4] של a?

_____

ה. הגדירו כעת את הפונקציה המחשבת את אורך החיתוך של שתי מחרוזות, s ו-t, לפי האלגוריתם הבא. הניחו שמבנה הנתונים הוא מערך דו-ממדי של שלמים, a, בגודל m*n, אורכי שתי מחרוזות הקלט. האיבר במקום ה-[i][j] במערך יציין את אורך החיתוך של תת-המחרוזת של s, המסתיימת במקום ה-i, עם תת-המחרוזת של t, המסתיימת במקום ה-j. למשל, עבור המחרוזות hello ו-there יכיל המערך את הנתונים הבאים:

	t	h	e	r	e
h	0	1	0	0	0
e	0	0	2	0	1
l	0	0	0	0	0
l	0	0	0	0	0
o	0	0	0	0	0

עבור המחרוזות pie ו-piece יכיל המערך את הנתונים הבאים:

	p	i	e	c	e
p	1	0	0	0	0
i	0	2	0	0	0
e	0	0	3	0	1

האלגוריתם מסתמך על האבחנה הבאה: אם s[i]==t[j], אזי a[i][j] מתקבל כתוצאה מהוספת 1 ל-a[i-1][j-1]. במילים אחרות, אורך החיתוך של תת המחרוזת של s המסתיימת במקום ה-i עם תת-המחרוזת של t המסתיימת במקום ה-j גדול ב-1 מאורך החיתוך של תתי המחרוזות המסתיימות במקומות i-1 ו-j-1, בהתאמה, אם s[i]==t[j]. במקרה ש-s[i]!=t[j], יהיה כמובן a[i][j]=0.
שימו לב שההגדרה תקפה רק אם i ו-j גדולים מ-0. את השורה העליונה ואת העמודה השמאלית של המערך a יש לאתחל מבלי להתבסס על תוצאת חישוב קודם.

הגדירו את הפונקציה intersect כדלהלן:
כותרת והגדרת משתנים;
הקצאת זיכרון עבור המערך a;
אתחלו את המערך כך שבמקום ה-[i][j] יהיה 1 אם s[i]==t[j], 0 אחרת;
עברו על המערך (פרט לשורה העליונה ולעמודה השמאלית ביותר) ועדכנו את ערכו של a[i][j] על פי a[i-1][j-1];
מצאו את הערך המקסימלי במערך: זהו אורך החיתוך המקסימלי, ואותו יש להחזיר.

ו. מהי סיבוכיות הזמן והמקום של הפונקציה `intersect` כפונקציה של `m` ו-`n`, אורכי המחרוזות `s` ו-`t`?

_____

_____

_____

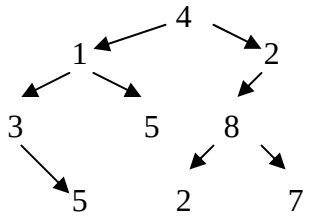
שאלה 3- 35 נקודות:

נתון עץ בינארי שכל צומת בו הוא רשומה מהטיפוס Tnode:

```
typedef struct tnode *TnodeP;
typedef struct tnode {
  int contents;
  TnodeP left, right;
} Tnode;
```

ניתן להניח כי העץ מכיל לפחות צומת אחד.

א. הגדירו פונקציה **רקורסיבית** המקבלת מצביע לעץ כזה ומחזירה את ערך האבר הגדול ביותר בעץ (כלומר, את ערך שדה contents המקסימלי בכל צומתי העץ). למשל, לעץ הבא תחזיר הפונקציה 8:



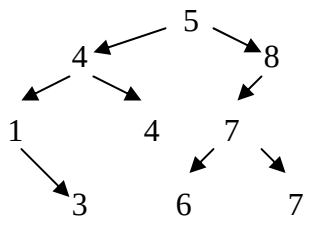
אם בעץ n צמתים, מהי סיבוכיות הפונקציה במונחים של n?

_____

_____

ב. הניחו כעת כי העץ מסודר כך שבכל צומת מתקיים התנאי הבא: אברי תת-העץ השמאלי של הצומת כולם קטנים מתוכן הצומת, ואברי תת-העץ הימני כולם גדולים או שווים לתוכן הצומת (ראו דוגמה לעץ כזה). הגדירו פונקציה רקורסיבית המקבלת מצביע לעץ כזה ומספר שלם `flag`, ומדפיסה את אברי העץ בסדר עולה אם `flag==0`, בסדר יורד אחרת.

דוגמה:



מהי סיבוכיות הפונקציה במונחים של `n`, מספר הצמתים בעץ?

_____

_____

פתרון:

.1

```
#include <stdio.h>


int f (int a[],int n) {
       int i,max;

       if (0==n) { /* if the array is empty */
         return 17;/* return an arbitrary number */
       } else {    /* otherwise, */
         max=a[0]; /* find the maximum element in the array */
         for (i=1;i<n;i++) {
           max=(a[i]>max)?a[i]:max;
         }         /* and return something greater than it */
         return max+1;
       }
}


int main()
{
        int array[]={23,12,3,6,45,22,8,10};
        printf("%d\n",f(array,sizeof(array)/sizeof(int)));

        return 0;
}
```

.2

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define ind(i,j) ((i)*n+(j))

int intersect (char *s, char *t);

int main(int argc,char **argv)
{
  if (argc!=3) {
    printf ("Usage: %s string1 string2\n", argv[0]);
    return 1;
  }
  printf("The longest intersection is of length %d\n",
         intersect(argv[1],argv[2]));
  return 0;
}

int *matalloc (int m, int n)
{
  return ((int *)malloc(m*n*sizeof(int)));
}
```

```
int intersect (char *s, char *t)
{
  int *a;
  int m=strlen(s), n=strlen(t);
  int i,j,max=0;

  if ((a=matalloc(m,n)) == NULL) {
    return -1;
  } else {
    for (i=0; i<m; i++) {
      for (j=0; j<n; j++) {
        a[ind(i,j)]=(s[i]==t[j]);
      }
    }
    max=0;
    for (i=1; i<m; i++) {
      for (j=1; j<n; j++) {
        if (a[ind(i,j)]) {
          a[ind(i,j)]+=a[ind(i-1,j-1)];
          if (a[ind(i,j)]>max) max=a[ind(i,j)];
        }
      }
    }
    return max;
  }
}
```

.3

```
#include <stdio.h>

typedef struct tnode *TnodeP;
typedef struct tnode {
        int contents;
        TnodeP left, right;
} Tnode;


int find_max(TnodeP r)
{
  int max_l, max_r, max=r->contents;

  if((r->left)==NULL && (r->right)==NULL) {
    max_l=max_r=max;
  } else if ((r->left)==NULL) {
    max_l=max_r=find_max(r->right);
  } else if((r->right)==NULL) {
    max_l=max_r=find_max(r->left);
  } else {
    max_l=find_max(r->left);
    max_r=find_max(r->right);
  }
  if (max_l>max) max=max_l;
  if (max_r>max) max=max_r;
  return max;
}
```

```
void sort(TnodeP root, int flag)
{
  if (root != NULL) {
    flag==0? sort(root -> left,flag): sort(root -> right,flag);
    printf("%d ", root->contents);
    flag==0? sort(root -> right,flag): sort(root -> left,flag);
  }
}


int main() {

        TnodeP root;
        Tnode p1,p2,p3,p4,p5,p6;

        p1.contents=18;
        p2.contents=20;
        p3.contents=15;
        p4.contents=12;
        p5.contents=19;
        p6.contents=11;

        root=&p3;
        p3.left=&p4;
        p3.right=&p5;
        p4.left=&p6;
        p4.right=NULL;
        p5.left=&p1;
        p5.right=&p2;
        p1.left=NULL;
        p1.right=NULL;
        p2.left=NULL;
        p2.right=NULL;
        p6.left=NULL;
        p6.right=NULL;

        printf("%d\n",find_max(root));
        sort(root,0);
        return 0;
}
```