מבוא למדעי המחשב
בחינת מועד ב', סמסטר א' תשס"ה, 13.2.2005

מרצה: שולי וינטנר.
מתרגל: עזרא דאיה.

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

1. ודאו כי בטופס שבידיכם 7 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	/30
2	/35
3	/35
סה"כ	/100

שאלה 1- 30 נקודות:

פרמוטציה של סדרת מספרים היא סידור כלשהו של כל אברי הסדרה. למשל, 3,0,6,5,2 היא פרמוטציה של הסדרה 5,6,0,2,3. הגדירו פונקציה המקבלת מערך של שלמים ואת אורכו, n, ומחזירה פרמוטציה **אקראית** של אברי המערך. המשמעות של פרמוטציה אקראית היא שבכל הפעלה של הפונקציה תתקבל פרמוטציה כלשהי של סדרת הקלט, כך שההסתברות לקבלת כל פרמוטציה היא שווה. ניתן להשתמש בפונקציה toss המקבלת מספר שלם k ומחזירה מספר שלם אקראי בין 0 ל-k-1. אין להשתמש באף פונקציה אחרת. סיבוכיות זמן: O(n). סיבוכיות מקום: O(1) (כלומר, על הפונקציה לסדר מחדש את אברי המערך שהיא מקבלת ללא שימוש במערך נוסף).

שאלה 2- 35 נקודות:

שתי מחרוזות, s ו-t, הן **דומות-n** אם ניתן לקבל את s על ידי מחיקת **בדיוק** n תווים מ-t. למשל, המחרוזות hell ו-hello הן דומות-1, המחרוזות bye ו-goodbye דומות-4, המחרוזות bb ו-ababa דומות-3 ואילו המחרוזות hello ו-there אינן דומות-n לאף ערך של n. מחרוזות זהות הן דומות-0.

א. הגדירו פונקציה בשם nsimilar המקבלת שתי מחרוזות ומספר שלם n ומחזירה 1 אם המחרוזות דומות-n, 0 אחרת. ניתן להשתמש בפונקציה strlen. מומלץ מאד להשתמש ברקורסיה.

מהי סיבוכיות הפונקציה במונחים של האורך של s, האורך של t ו-n?

_____

_____

_____

ב. הגדירו תוכנית המקבלת משורת הפקודה שתי מחרוזות ומדפיסה את ערכו של n כך שהמחרוזות הן n-דומות, או 1- אם לא קיים אף n כזה. ניתן להשתמש בפונקציה nsimilar מהסעיף הקודם גם אם לא מימשתם אותה. על התוכנית לבדוק שמספר הארגומנטים שלה תקין.

מהי סיבוכיות הפונקציה במונחים של האורך של s, והאורך של t?

_____

_____

_____

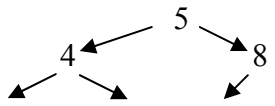
שאלה 3- 35 נקודות:

נתון עץ בינארי שכל צומת בו הוא רשומה מהטיפוס Tnode:

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

עץ הוא **ימני** אם בכל צומת מתקיים התנאי הבא: אברי תת-העץ השמאלי של הצומת כולם קטנים מתוכן הצומת, ואברי תת-העץ הימני כולם גדולים או שווים לתוכן הצומת. עץ הוא **שמאלי** אם בכל צומת מתקיים התנאי הבא: אברי תת-העץ הימני של הצומת כולם קטנים מתוכן הצומת, ואברי תת-העץ השמאלי כולם גדולים או שווים לתוכן הצומת.

לדוגמה, עץ ימני:



עץ שמאלי:

א. הגדירו פונקציה **רקורסיבית** המקבלת מצביע לעץ ימני ומחזירה את ערך האבר הגדול ביותר בעץ (כלומר, את ערך שדה `contents` המקסימלי בכל צומתי העץ). ניתן להניח כי העץ מכיל לפחות צומת אחד.

ב. הגדירו פונקציה רקורסיבית בשם `insert` המקבלת מצביע לעץ ימני ומספר שלם `n`, ומוסיפה צומת שתוכנו `n` לעץ כך שהעץ המתקבל נשמר עץ ימני. על הפונקציה להחזיר מצביע לעץ החדש.

ג. הגדירו פונקציה המקבלת מצביע לעץ שמאלי ומחזירה (דרך שורת הפרמטרים שלה) מצביע לעץ ימני שמכיל את אותם נתונים בדיוק. אין לפגוע בעץ השמאלי הנתון. ניתן להשתמש בפונקציה `insert` של הסעיף הקודם גם אם לא מימשתם אותה.

Number2 :

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

int nsimilar (char *s, char *t, int n);

int main(int argc,char **argv)
{
  int i;
  if (argc!=3) {
    printf ("Usage: %s string1 string2\n", argv[0]);
    return 1;
  }
  i=strlen(argv[2])-strlen(argv[1]);
  while (i>=0 && !nsimilar(argv[1],argv[2],i)) {
    i--;
  }
  printf("The strings \"%s\" and \"%s\" are %d-similar\n",
              argv[1],argv[2],i);
  return 0;
}

int nsimilar (char *s, char *t, int n)
{
  if (s[0]=='\0' && t[0]=='\0') {
    return (n==0);
  } else if (s[0]=='\0') {
    return (n==strlen(t));
  } else if (t[0]=='\0') {
    return 0;
  } else if (s[0]==t[0]) {
    return nsimilar(s+1,t+1,n);
  } else {
    return nsimilar(s,t+1,n-1);
  }
}
```

Number3:

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

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

int find_max(TnodeP root);
```

```
TnodeP insert(TnodeP root, int n);
void left_to_right(TnodeP root, TnodeP *right_tree);


int find_max(TnodeP root)
{
  if(root->right == NULL) {
    return root->contents;
  } else {
    return find_max(root->right);
  }
}

TnodeP insert(TnodeP root, int n)
{
  TnodeP p;

  if(root==NULL) /* new node has to be added */ {
    p=(TnodeP)malloc(sizeof(Tnode));
    if (p!=NULL) {
      p->contents=n;
      p->right=NULL;
      p->left=NULL;
    }
    return p;
  }
  if (n >= root->contents) { /* add to the right */
    root->right = insert(root->right,n);
  } else { /* add to the left */
    root->left = insert(root->left,n);
  }
  return root;
}

void left_to_right(TnodeP root,TnodeP *right_tree)
{
  if (root != NULL) {
    *right_tree=insert(*right_tree,root->contents);
    left_to_right(root -> left,right_tree);
    left_to_right(root -> right,right_tree);
  }
}


void print_tree (TnodeP p)
{
  if (p != NULL) {
    print_tree (p->left);
    printf ("%4d\n", p->contents);
    print_tree (p->right);
```

```
  }
}

int main() {

        TnodeP root,right_tree=NULL;
        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;
*/
        p1.contents=12;
        p2.contents=8;
        p3.contents=15;
        p4.contents=19;
        p5.contents=10;
        p6.contents=22;

        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;

        print_tree(root);
        printf("%d\n",find_max(root));
        left_to_right(root,&right_tree);
        print_tree(right_tree);
        printf("%d\n",find_max(right_tree));

        return 0;
}
```