מבוא למדעי המחשב
בחינת מועד ב', סמסטר ב' תשס"ה, 22.7.2005

מרצה: גב' יעל כהן-סיגל.
מתרגל: מר עזרא דאיה.

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

1. יש לענות על כל השאלות.
2. קראו היטב כל שאלה. ודאו כי אתם מבינים את השאלה לפני שתתחילו לענות עליה.
3. כתבו בכתב יד ברור וקריא. תשובות לא קריאות לא תיבדקנה.
4. הערות לתשובותיכם ניתן לכתוב בעברית, גם בגוף פונקציות C.
5. ניתן ונדרש להגדיר פונקציות עזר לפי הצורך.
6. ניתן להשתמש בכל פונקציה המופיעה במצגות ההרצאות והתרגולים ע"י הצהרה עליה בלבד (אין צורך להגדירה). כמו כן, ניתן להשתמש בפונקציות מתוך הספריות stdio.h, stdlib ו-string.h. לא ניתן להשתמש בפונקציות אחרות בלא להגדיר אותן במפורש.

שאלה 1 (25 נק')

כתבו פונקציה המקבלת מערך של מספרים שונים זה מזה ואת גודלו ומסדרת את אברי המערך כך שיתקיימו כל התנאים הבאים:

1. כל האיברים באינדקסים הזוגיים מהווים סדרה יורדת, כלומר $arr[0] > arr[2] > arr[4] > \ldots$.
2. כל האיברים באינדקסים האי-זוגיים מהווים סדרה עולה, כלומר $arr[1] < arr[3] < arr[5] < \ldots$.
3. כל איבר באינדקס אי-זוגי קטן מהאיברים הסמוכים לו, כלומר לכל אינדקס אי-זוגי i במערך $arr[i-1] > arr[i] < arr[i+1]$.
4. כל איבר באינדקס זוגי גדול מהאיברים הסמוכים לו, כלומר לכל אינדקס זוגי i במערך $arr[i-1] < arr[i] > arr[i+1]$.

למשל,

עבור המערך 11 , 12 , 35 , 1 , 3 , 9 , 18
הסידור הרצוי הוא 11 , 9 , 12 , 3 , 18 , 1 , 35

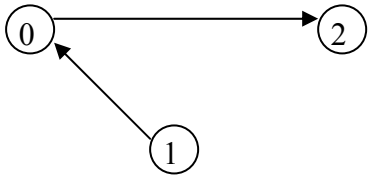
עבור המערך 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1
הסידור הרצוי הוא 5 , 4 , 6 , 3 , 7 , 2 , 8 , 1 , 9

דרישות סיבוכיות:
סיבוכיות זמן: O(n log n) סיבוכיות מקום: O(n)
פתרונות בסיבוכיות גבוהה יותר לא יתקבלו.

שאלה 2 (25 נק')

גרף מכוון G הוא זוג (V,E) כאשר V היא קבוצת הקודקודים ו- E הינה קבוצת הקשתות (כאשר קשת הינה זוג סדור (i, j), שמשמעותה היא שקיימת קשת היוצאת מן הקודקוד ה- i ונכנסת אל הקודקוד ה- j).

למשל: הגרף $G = (\underbrace{\{0,1,2\}}_{V}, \underbrace{\{(0,2),(1,0)\}}_{E})$ הוא בעל הייצוג:



מסלול בגרף מכוון G=(V,E) הינו סדרה של קשתות $(i_1,i_2),(i_2,i_3),(i_3,i_4),...,(i_{n-1},i_n)$ כך ש $i_1,i_2,i_3,i_4,...,i_{n-1},i_n \in V$ וגם $\{(i_1,i_2),(i_2,i_3),(i_3,i_4),...,(i_{n-1},i_n)\} \subseteq E$. למשל, בגרף לעיל קיימים מסלולים מקודקוד 1 לקודקוד 2 , מקודקוד 1 לקודקוד 0 וכן מקודקוד 0 לקודקוד 2.

גרף מכוון G=(V,E) נקרא ללא מעגלים אם לא קיימים בו קודקודים i,j כך שקיים מסלול מ- i ל- j וגם קיים מסלול מ- j ל-i.

שימו לב כי לכל קודקוד מוצמד מספר סידורי הייחודי לו. לצרכי שאלה זאת, נניח כי קודקודי הגרף ממוספרים בסדר עולה החל מאפס (כלומר, אם |V|=n אזי V={0,1,…,n-1}).

ניתן לייצג גרף ע"י מטריצת שכנויות בינארית g, כאשר

$$g[i, j] = \begin{cases} 0 & (i, j) \notin E \\ 1 & (i, j) \in E \end{cases}$$

למשל, הגרף לעיל ייוצג ע"י המטריצה

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

כתבו פונקציה רקורסיבית המקבלת כקלט מטריצת שכנויות המייצגת גרף מכוון ללא מעגלים כלשהוא, את מספר הקודקודים בגרף וכן שני קודקודים כלשהם בגרף ומחזירה 1 באם קיים מסלול בגרף מן הקודקוד הראשון לשני, אחרת תחזיר הפונקציה 0.

הערות:

1. ניתן להניח כי מטריצת השכנויות אכן מייצגת גרף מכוון ללא מעגלים (אין צורך לבדוק זאת).
2. ניתן להניח כי גודל המטריצה מוגדר כקבוע (ע"י define).

שאלה 3 (50 נק')

נתונות הרשומות הבאות:

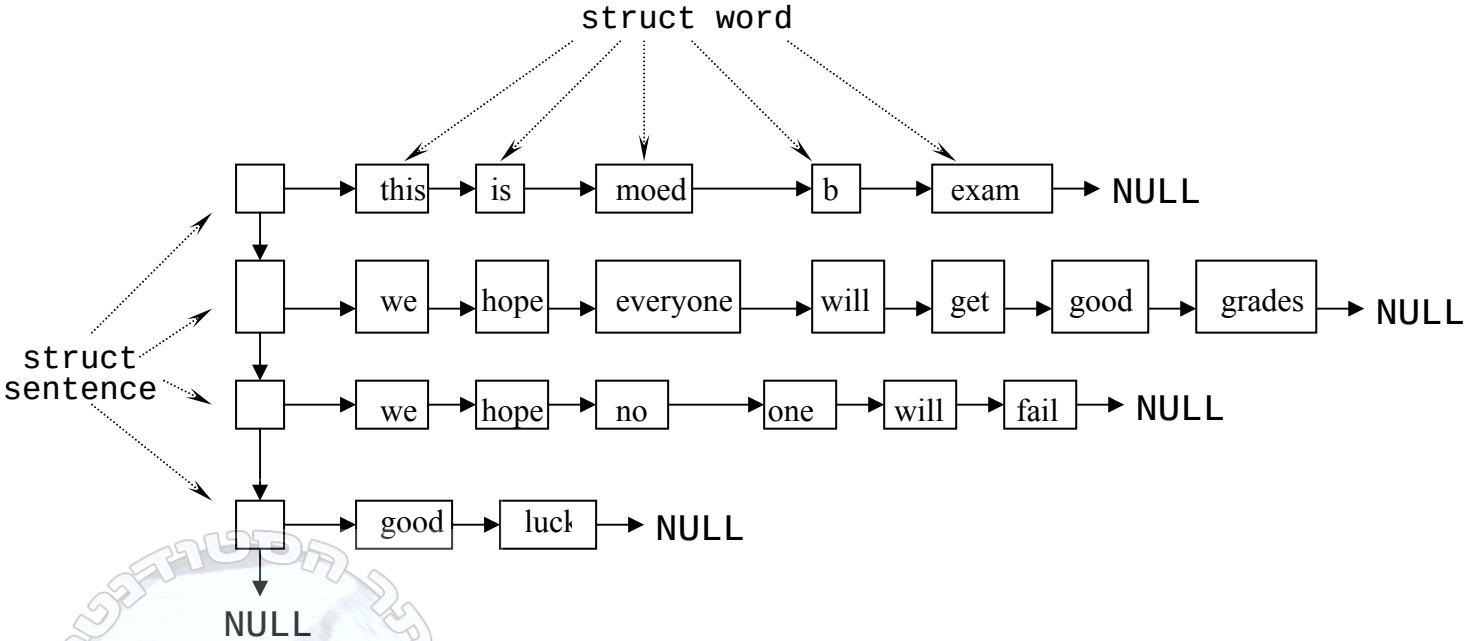
```
struct word {
        char *s;
        struct word *next;
};

struct sentence {
        struct word *wordlist;
        struct sentence *next;
};
```

בעזרת רשומות אלו נייצג רשימת משפטים כאשר כל משפט הינו אוסף של מילים.

למשל, רשימת המשפטים:

this is moed b exam
we hope everyone will get good grades
we hope no one will fail
good luck

תיוצג באופן הבא:



א. כתבו פונקציה המקבלת מצביע לרשימת משפטים (כלומר מצביע ל- struct sentence) ומילה (המיוצגת כמחרוזת) ומדפיסה את כל המשפטים בהם מופיעה המילה. אין להדפיס משפט יותר ממספר הפעמים בהם הוא מופיע ברשימה.
למשל, עבור קלט המייצג את רשימת המשפטים

Good luck
good luck
good luck
this is moed b exam

והמילה good יודפסו המשפטים הבאים:

good luck
good luck

ב. כתבו פונקציה המקבלת מצביע לרשימת משפטים ומדפיסה את כל המשפטים המופיעים ברשימה יותר מפעם אחת.
הערה: ניתן להדפיס יותר מפעם אחת משפט המופיע יותר מפעם אחת אך לא יותר ממספר המופעים שלו ברשימה.
למשל, עבור קלט המייצג את רשימת המשפטים

Good luck
good luck
good luck
this is moed b exam

יודפס המשפט

good luck

<u>בהצלחה!</u>

Number1:

```
#include <stdio.h>
#define SIZE 100

enum {FALSE,TRUE};

void mergeSort (int array[], int size);
int  merge  (int list1[] , int size1 , int list2[] , int size2 , int list3[]);

void arrange (int arr[] , int len)
{
	int tmp[SIZE], i, j;
	mergeSort(arr,len);
	for (i=0 , j=0  ; i<len; i+=2 , j++){
		tmp[i]=arr[len-1-j];
		tmp[i+1]=arr[j];
	}
	if (len%2==0){
		tmp[i]=arr[len-1];
	}
	for (i=0; i< len; i++)
		arr[i]=tmp[i];
	return;
}
```

Numver 2:

```
#include <stdio.h>
#define SIZE 100

enum {FALSE,TRUE};

int FindPath (int g[SIZE][SIZE] , int v, int s, int t)
{
	int i;
	if (g[s][t]==TRUE)
		return TRUE;
	for (i=0; i<v; i++){
		if (g[s][i]==TRUE && FindPath(g,v,i,t)==TRUE)
			return TRUE;
	}
	return FALSE;
}
```

Number 3:

```
#include <stdio.h>
#include <string.h>
#define LEN 100

enum {FALSE,TRUE};

struct word {
	char *s;
	struct word *next;
};

struct sentence {
	struct word *wordlist;
	struct sentence *next;
};

void FindWord (struct sentence *list , char *s)
{
	struct word *tmp;
	for ( ; list!=NULL; list=list->next){
		for (tmp=list->wordlist; tmp!=NULL; tmp=tmp->next){
			if (strcmp(tmp->s,s)==0){
				PrintSentence (list);
				break;
			}
		}
	}
	return;
}


void PrintSentence (struct sentence *list)
{
	struct word *tmp;
	for (tmp=list->wordlist; tmp!=NULL; tmp=tmp->next)
		printf("%s ",tmp->s);
	printf("\n");
	return;
}

void printDuplicate (struct sentence *list)
{
	struct sentence *s;
	struct word *w1,*w2;
	int flag;
	for ( ; list->next!=NULL; list=list->next){
		for (s=list->next; s!=NULL; s=s->next){
			flag=TRUE;
```

```
                    for (w1=list->wordlist , w2=s->wordlist; w1!=NULL &&
w2!=NULL; w1=w1->next, w2=w2->next){
                        if (strcmp(w1->s,w2->s)!=0){
                                flag=FALSE;
                                break;
                        }
                    }
                    if (flag==TRUE && w1==NULL && w2==NULL){
                        PrintSentence(list);
                        break;
                    }
                }
        }
        return;
}
```