מבוא למדעי המחשב
מועד מיוחד, סמסטר ב' תשס"ה, 16.8.2005

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.h ו- string.h. לא ניתן להשתמש בפונקציות אחרות בלא להגדיר אותן במפורש.

שאלה 1 (30 נק')

כתבו פונקציה המקבלת שני מערכים של מספרים שלמים ואת גודלם ומדפיסה את כל המספרים המופיעים רק באחד מן המערכים (כלומר, את כל המספרים המופיעים במערך הראשון אך לא בשני וכן את כל המספרים המופיעים במערך השני אך לא בראשון).

לדוגמא:

עבור המערכים 23,4,-3,5,6 ו- 1,23,5,7,-3 יודפסו המספרים 1,7,4,6 (המספרים 4 ו- 6 יודפסו כיון שהם מופיעים במערך הראשון אך לא בשני, והמספרים 1 ו- 7 יודפסו כיון שהם מופיעים במערך השני אך לא בראשון).

אין חשיבות לסדר הדפסת המספרים.

דרישות סיבוכיות:
סיבוכיות זמן: O(n log n) סיבוכיות מקום: O(n)
כאשר n הוא גודל המערך הגדול מבין השניים.
פתרונות בסיבוכיות גבוהה יותר לא יתקבלו.

שאלה 2 (30 נק')

מסלול במערך דו-מימדי בגודל $SIZE \times SIZE$ הינו סדרה של תאים סמוכים (לא באלכסון) המכילים ערך זהה והמתחילה מהפינה השמאלית העליונה ומסתיימת בפינה השמאלית התחתונה.

לדוגמא:

המערך

1	2	3	1	1	1	4
1	0	0	1	3	1	5
1	1	1	1	4	1	9
9	4	9	9	9	1	9
3	4	0	1	1	1	7
2	1	0	1	0	0	8
1	1	0	1	1	1	1

מכיל מסלול (של 1-ים).

המערך

4	4	1	6
1	4	2	7
2	4	3	8
3	4	7	4

אינו מכיל מסלול.

שימו לב כי מכל תא ניתן לנוע לתא סמוך בכיוון אופקי או אנכי אך לא באלכסון.
כתבו פונקציה רקורסיבית המקבלת מערך דו מימדי בגודל $SIZE \times SIZE$ (ניתן להניח כי SIZE מוגדר ע"י define) ומחזירה 1 באם קיים במערך מסלול, אחרת תחזיר הפונקציה 0.

שאלה 3 (40 נק')

נתונה רשימה מקושרת אשר כל תא בה מוגדר באופן הבא:

```
struct cell{
    const int num;
    struct cell *next;
};
```

הגדירו פונקציה המקבלת מצביע לתחילת רשימה מקושרת כנ"ל, ממיינת את איבריה בעזרת אלגוריתם המיון Insertion Sort ומחזירה מצביע לתחילת הרשימה ממוינת.
שימו לב: השדה num ברשומה cell מוגדר כ- const ולכן לא ניתן לשנותו!.

סיבוכיות נדרשת:
סיבוכיות זמן: $O(n^2)$ (n הוא מספר האיברים ברשימה)
סיבוכיות מקום: O(1)

בהצלחה!

Number 2:

```
#include <stdio.h>
#define N 4

enum {FALSE, TRUE};

void solve (int board[N][N])
{
	int visit[N][N]={0};
	if (board[0][0]!=board[N-1][N-1])
		printf("0\n");
	else
		printf("%d\n",recsolve(board,visit,0,0));
	return;
}

int recsolve (int board[N][N], int visit[N][N], int r, int c)
{
	visit[r][c]=TRUE;
	if ( (r == N-1) && (c==N-1) ) {
		return TRUE;
	}
	if ( (r>0) && (board[r-1][c]==board[0][0]) && (visit[r-1][c]==FALSE) &&
(recsolve(board,visit,r-1,c)==TRUE) )
				return TRUE;
	if ( (r<N-1) && (board[r+1][c]==board[0][0]) && (visit[r+1][c]==FALSE)
&& (recsolve(board,visit,r+1,c)==TRUE) )
				return TRUE;
	if ( (c>0) && (board[r][c-1]==board[0][0]) && (visit[r][c-1]==FALSE) &&
(recsolve(board,visit,r,c-1)==TRUE) )
				return TRUE;
	if ( (c<N-1) && (board[r][c+1]==board[0][0]) && (visit[r][c+1]==FALSE)
&& (recsolve(board,visit,r,c+1)==TRUE) )
				return TRUE;
	visit[r][c]=FALSE;
	return FALSE;
}
```

Number 3:

```
#include <stdio.h>

#define SIZE 100

struct cell{
	const int num;
	struct cell *next;
```

```
};

struct cell  *sort  ( struct cell *list )
{
	struct cell *current, *prevcurrent, *tmp, *prevtmp;
	/* empty list or only one element */
	if ( (list==NULL) || (list->next==NULL) )
		return list;
	/* more than one element in the list */
	prevcurrent=list;
	current=list->next;
	while (current!=NULL){
		/*find place to insert current*/
		if(list->num > current->num){ /* at the beginning */
			prevcurrent->next=current->next;
			current->next=list;
			list=current;
			current=prevcurrent->next;
		}
		else{ /* not at the beginning */
			for (prevtmp=list, tmp=list->next; tmp->num<current-
>num; tmp=tmp->next, prevtmp=prevtmp->next);
			if (tmp!=current){
				prevcurrent->next=current->next;
				prevtmp->next=current;
				current->next=tmp;
				current=prevcurrent->next;
			}
			else{
				prevcurrent=current;
				current=current->next;
			}
		}
	}
	return list;
}
```