מבוא למדעי המחשב מ' 234114

מבחן מועד ב', סמסטר אביב תשס"ב 02.10.02

שם משפחה	שם פרטי	מס' סטודנט

משך המבחן: 3 שעות.
חומר עזר: אין להשתמש בחומר עזר, מלבד הדף שחולק.

שאלה	ערך	הישג
1	15	
2	15	
3	12	
4	11	
5	16	
6	18	
7	15	
סכום	102	

הוראות לנבחנים ולנבחנות:

1. מלאו את הפרטים בראש דף זה (בעט).
2. בדקו שיש 12 עמודים (7 שאלות) כולל עמוד זה.
3. התשובות ייכתבו על טופס המבחן.
4. כתבו בכתב-יד נקי וברור (מומלץ להשתמש בעפרון ומחק).
5. אין לכתוב הערות והסברים לתשובות.
6. בכל השאלות ניתן להניח שהקלט תקין.

מרצה: רוני למפל

מתרגלים: רועי מלמד, אורית כהן,שגיא שיין.

<u>שאלה 1 (15 נקודות)</u>

נתונות שתי הפונקציות הבאות:

```
double f (unsigned int num);
double g (unsigned int num);
```

ידוע כי f() הינה פונקציה עולה ממש, וכי g() הינה פונקציה יורדת ממש (לכל מספר טבעי n, f(n)<f(n+1) ואילו g(n)>g(n+1)).

א. השלימו את הפונקציה intersect_point, המקבלת שלם אי שלילי range, ומחפשת מספר <u>שלם</u> num בתחום [0,range] עבורו f(num) == g(num). אם יש מספר כזה, על הפונקציה להחזירו. אם אין מספר כזה, הפונקציה תחזיר range+1.
על הפונקציה להיות יעילה ככל האפשר.

```
unsigned int intersect_point ( unsigned int range )
{

}
```

ב. מהי סיבוכיות הזמן והזיכרון של הפונקציה שכתבתם? הניחו כי כל קריאה לפונקציות f,g דורשת **O**(1) זמן וזיכרון.

שאלה 2 (15 נקודות)

בשאלה זו נתייחס למימוש המחסנית שהוצג בהרצאות, ונרחיב את הדוגמה כבתרגיל בית 6.

א. [**9 נקודות**] בסעיף זה הנכם מתבקשים לממש את הפעולה st_push עבור מחסנית אשר גודלה המירבי אינו מוגבל . נתונה לכם דוגמת מימוש של הפונקציה st_push עבור מחסנית שגודלה נקבע בזמן ההקצאה. עליכם לממש את אותה הפונקציה עבור מחסנית שגודלה משתנה בהתאם לתכולתה. להזכירכם כאשר המחסנית מלאה עליכם להגדיל את המחסנית פי 2 מגודלה הנוכחי (תוך שמירה על תכולתה).

```
typedef struct stack {
  double  *       array;
  unsigned int  top;
  unsigned int  alloc_size;
 } stack_t;

typedef
enum { ST_NULL_STACK = -1, ST_OK, ST_MEMORY, ST_EMPTY, ST_OVERFLOW }
st_code_t;

st_code_t  st_push (stack_t * stack,double obj)  {
     if ( !stack ) return ST_NULL_STACK;
     if ( stack->top == stack->alloc_size ) return ST_OVERFLOW;
     stack->array[stack->top++] = obj;
     return ST_OK;
 }
```

רשמו את הפתרון כאן :

```
st_code_t  st_push (stack_t * stack,double obj)
{



}
```

ב. [**6 נקודות**] נתון לכם המימוש של הפונקציה st_alloc() . כתבו את המימוש של הפונקציה st_free() .

```
stack_t* st_alloc(int size)
{
   stack_t* stack = (stack_t*)malloc(sizeof(stack_t));
   if(stack)
   {
        stack->array = (double*)malloc(sizeof(double)*size);
        if(!stack->array)
        {
          free(stack);
          return NULL;
         }
        stack->top = 0;
        stack->alloc_size = size;
     }
   return stack;
 }
```

רשמו את הפתרון כאן

```
void st_free(stack_t* st)
 {

}
```

<u>שאלה 3 (12 נקודות)</u>

עליכם להשלים את הפונקציה partial_sums, שהצהרתה מובאת להלן:

```
#define N (20)
void partial_sums(int arr[N], int sums[N][N]);
```

הפונקציה מקבלת מערך שלמים arr באורך N ומטריצת שלמים sums שמימדיה NxN. לכל זוג אינדקסים i,j המקיימים $0 \le i \le j < N$, על הפונקציה למלא ב- sums[i][j] וב- sums[j][i] את סכום האיברים שבתאים ה- i עד ה- j (כולל) במערך arr.

סיבוכיות זמן נדרשת: $O(N^2)$. סיבוכיות מקום נוסף: O(1) .

```
void partial_sums(int arr[N], int sums[N][N])
{

}
```

שאלה 4 (11 נקודות)

השלימו את הקוד הבא במקומות המסומנים, כך שהפונקציה ()my_sort תמיין את המערך arr (שאורכו len) בסדר לא-יורד.
אין להגדיר משתנים או פונקציות נוספות, ואין לשנות את הפונקציה ()my_swap .

```
void my_swap(int arr[], int place)
{
    int  temp  = arr[0];
    arr[0]     = arr[place];
    arr[place] = temp;
}

void f(int arr[], int len)
{
    int j;

    for ( _________ ; _______________ ; ___________ )


          if ( _____________ < _________________ )


                 my_swap( arr,  _____________ );
}

void my_sort(int arr[], int len)
{
    int k;

    for ( _________ ; _______________ ; ___________ )


          f ( __________________ , ___________________);
}
```

<u>שאלה 5 (16 נקודות)</u>

מוגדר הטיפוס הבא:

typedef enum { FALSE, TRUE } Boolean;

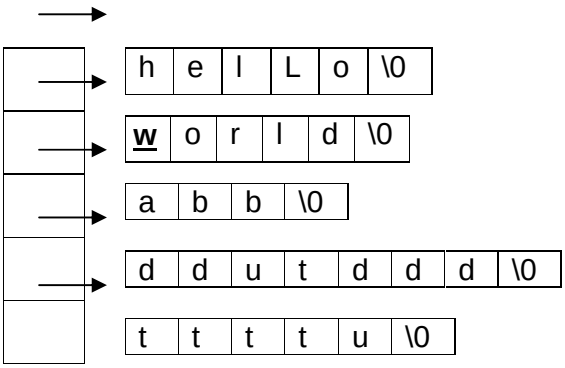
נתון מערך בגודל n של מחרוזות, בו כל מחרוזת מכילה אותיות אנגליות קטנות בלבד.

נסמן את המחרוזת שבמקום ה- k במערך ב- str_k.

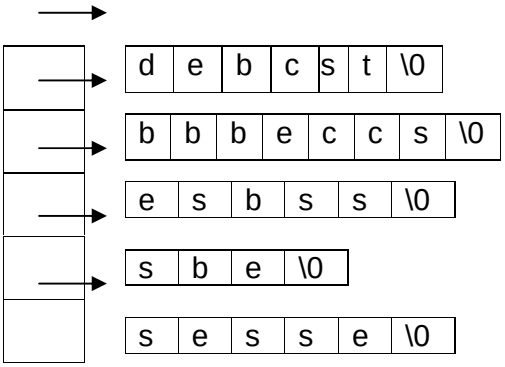
מערך כזה יקרא *<u>מערך טלסקופי</u>*, אם **לכל זוג אינדקסים** m,k המקיימים $0 \le m < k < n$, **<u>כל</u>** האותיות המרכיבות את str_k מופיעות גם ב- str_m. במילים אחרות, כל אחת מהאותיות במחרוזת ה- k-ית מופיעה בכל אחת מהמחרוזות הקודמות למחרוזת ה-k-ית במערך.

<u>דוגמא</u> : נתונים שני המערכים הבאים:

מערך 1



מערך 2



מערך 1 אינו מערך טלסקופי (למשל האות w המופיעה במחרוזת במקום ה-1 במערך אינה מופיעה במחרוזת במקום ה-0 במערך). מערך 2 הינו מערך טלסקופי.

כתבו את הפונקציה check_telescopic_array, המקבלת את מערך המחרוזות ואת גודלו.

הפונקציה תחזיר TRUE אם המערך אותו קבלה הוא טלסקופי (אחרת יוחזר FALSE).

על הפונקציה לעבוד בסיבוכיות זמן לינארית בסכום אורכי המחרוזות שבמערך.

```
Boolean  check_telescopic_array (unsigned size, char* string_array[])
{

}
```

שאלה 6 (18 נקודות)

בשאלה זו נגדיר פעולה H המשנה מערך arr של משתני double באופן מסוים.
H פועלת רק על מערכים שאורכם הוא חזקה שלמה של 2, כלומר על מערכים באורך 1,2,4,8 וכו'.

כאשר arr באורך 1, H אינה משנה אותו.
עבור מערך arr באורך 2, נסמן arr[0]=x ו- arr[1]=y.
H תשנה את arr כך שיתקיים arr[0] = x+y , arr[1] = x-y .

נניח כי ידוע כיצד לחשב את H עבור מערך באורך 2^k, ונתאר כיצד לחשב את H עבור מערך arr באורך 2^{k+1}. נסמן את חציו הראשון (2^k האיברים הראשונים) ב- arr_left, ואת חציו השני ב- arr_right.
ראשית, יש להפעיל על כל חצי את H (הנחנו כי ידוע כיצד H פועלת על מערכים באורך 2^k). כעת, נשנה את החצי הראשון של arr כך שיכיל את H(arr_left)+H(arr_right), ונשנה את החצי השני של arr כך שיכיל את H(arr_left)-H(arr_right). פעולות החיבור והחיסור מתבצעות איבר איבר, כבחיבור וחיסור וקטורים. שימו לב שתיאור זה מתאים גם להגדרת arr על מערכים באורך 2.

נדגים את התהליך עבור מערך arr באורך 4 המכיל את האיברים a,b,c,d בארבעת תאיו.
על מנת לחשב את H(arr), נשים לב כי arr_left={a,b} וכי arr_right = {c,d}.
לפי הגדרת H עבור מערכים באורך 2,

H(arr_left) = { a+b, a-b } ו- H(arr_right) = { c+d, c-d } .

לכן, H(arr) = { (a+b)+(c+d), (a-b)+(c-d), (a+b)-(c+d), (a-b)-(c-d) } .

א. (12 נקודות) השלימו את הפונקציה הרקורסיבית calcH, הקולטת מערך של משתני double ואת אורכו, ומפעילה את H על המערך. ניתן להניח כי אורך המערך הוא חזקה שלמה של 2.

```
void calcH (double arr[], unsigned int len)
{
unsigned int i, mid = len / 2;

if ____________________________________________

____________________________________________

____________________________________________

for ( i = 0; i < mid; i++ ) {

____________________________________________

____________________________________________

____________________________________________
}
}
```

ב. (3 נקודות) מהי סיבוכיות הזכרון הנוסף של calcH (כפונקציה של אורך המערך len)? נמקו.

ג. (3 נקודות) מהי סיבוכיות הזמן של calcH? נמקו.
הדרכה: למדתם אלגוריתם רקורסיבי נוסף בו מבנה הקוד זהה (הפונקציה הרקורסיבית מבצעת את אותו מספר קריאות רקורסיביות ואותו סדר גודל של עבודה נוספת כמו calcH). סיבוכיות הזמן של אלגוריתם זה הינה זהה לזו של calcH .

<u>שאלה 7 (15 נקודות)</u>

נתונות ההגדרות הבאות:

```
#define ALPHABET (4)
#define N (5)
```

כתבו תוכנית המדפיסה את כל הוקטורים באורך N*ALPHABET שבהם **כל אחת** מהספרות 0,…ALPHABET-1 מופיעה **בדיוק** N פעמים.

לדוגמה, עבור ההגדרות שניתנו, אחד הוקטורים הנדרשים הינו 01233210332211003012 .

<u>הנחיות</u>:

1. על הפונקציה ()main להקצות משתנים, לאתחל אותם, ולקרוא לפונקציה הבונה את הוקטורים, ששמה יהיה ()build_vectors.
2. על הפונקציה ()build_vectors לקלוט את גודל האלפבית כפרמטר, אסור לה להשתמש בקבוע ALPHABET .
3. אין להשתמש במשתנים גלובליים או סטטיים.
4. הניחו כי לרשותכם עומדת הפונקציה void print_array(int arr[], int len) המדפיסה מערך באורך len של משתני int .
5. כיתבו את פונקצית ה- ()main בעמוד זה, ואת הפונקציה ()build_vectors בעמוד הבא.

```
int main()
{

}
```

ב. (8 נקודות) כעת נרצה להעתיק את הסכומים החלקיים שבמערך sums לשטח חדש, אולם לא נרצה להעתיקם למערך דו-מימדי חדש, מכיוון שאין ברצוננו לשמור ערכים כפולים.

השלימו את הפונקציה הבאה, המקצה באופן דינאמי N מצביעים ו- N(N+1)/2 משתנים מטיפוס int, ומארגנת את המצביעים לשטח המשתנים כך שהקוד המעתיק את המרחקים מ- sums לשטח החדש (שלוש השורות הכתובות בסוף הפונקציה), פועל כשורה.
על הפונקציה להחזיר 0 אם ההקצאה הדינאמית הצליחה, ו- 1 אם היא נכשלה.

הערה: 1+2+… +N = N(N+1)/2

```
#include <stdlib.h>

int copy_sums(int sums[N][N])
{
int ** rows = NULL;
int *  data = NULL;
int    i,j;

/* first, allocate the pointers and the area for the int variables */

if ( !(rows = (_____) malloc (_____ * sizeof(_____)) ) ||

!(data = (_____) malloc (_____ * sizeof(_____)) )     )
{

_____

_____
}

/* organize the pointers */

rows[0] = _____

for ( j = 1; j < N; j++ )

rows[ j ] = _____

/* now we can safely copy the data  to the new area */
for ( i = 0; i < N; i++ )
for ( j = 0; j <= i; j++ )
rows[ i ][ j ] = sums[ i ][ j ];

_____

}
```