



Today:

---

***Summary -  
Examples  
from exams***



# Example – strings 2006/a 2 q2-a

---

- סעיף א' (12 נקודות):

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
int chr_count(char *str, char ch);
```

הפונקציה תחזיר את מספר ההופעות של `ch` במחרוזת `.str`.

- הערה: אין להשתמש בפונקציות מתוך הספרייה `.string.h`.



# Example – strings 2006/a 2 q2-a

---

## ■ דוגמאות:

(1) התו **a** מופיע **3** פעמים במחרוזת **cbadaca**

(2) התו **g** מופיע **0** פעמים במחרוזת **cbadaca**



# Solution – strings 2006/a 2 q2-a

---

```
int chr_count(char *str, char ch) {  
    int count=0;  
    while(*str != '\0') {  
        if (*str == ch)  
            count++;  
        str++;  
    }  
    return count;  
}
```



# Example – strings 2006/a 2 q2-b

---

■ אנגרמה - מחרוזת str2 נקראת "אנגרמה" של מחרוזת str1 אם ניתן לקבל את str2 ע"י שינוי סדר התווים (פרמוטציה) של str1.

- דוגמאות:
- (1) המחרוזת "orchestra" היא אנגרמה של המחרוזת "carthorse"
  - (2) המחרוזת abcd איננה אנגרמה של המחרוזת abcda
  - (3) המחרוזת abcda איננה אנגרמה של המחרוזת abcdb



# Example – strings 2006/a 2 q2-b

---

כתבו פונקציה בעלת המפרט (prototype) הבא:

```
int is_anagram(char *str1, char *str2);
```

הפונקציה תחזיר 1 אם המחרוזת str2 היא אנגרמה של המחרוזת str1, ואחרת היא תחזיר 0.



# Solution – strings 2006/a 2 q2-b

---

```
int is_anagram(char *str1, char *str2) {
    int i;
    for(i=0; str1[i] != '\0'; i++)
        if (chr_count(str1, str1[i]) != chr_count(str2, str1[i]))
            return 0;
    if (str2[i] != '\0') /* Different lengths */
        return 0;
    return 1;
}
```



## Example– lists 2010/a 2 q3

---

- בשאלה זו נבנה מערכת פשוטה המאפשרת חישובים גיאומטריים. בבסיס המערכת נגדיר ייצוג של נקודה במרחב בעזרת קוארדינטות ה-x וה-y שלה. נתון המבנה Point המגדיר נקודה כזו.

```
typedef struct point_t  
{  
    double x, y;  
} Point;
```





# Example— lists 2010/a 2 q3-a

---

## ■ סעיף א' (5 נקודות):

- הגדירו מבנה בשם Triangle המייצג משולש במישור. על המבנה להחזיק את ערכי שלושת הקודקודים המגדירים את המשולש וכן שדה נוסף שיאפשר יצירת רשימה מקושרת של משולשים.



# Solution– lists 2010/a 2 q3-a

---

```
typedef struct triangle
{
    point_t a, b, c;
    struct triangle* next;
} triangle_t;
```



# Example— lists 2010/a 2 q3-b

■ סעיף ב' (8 נקודות):

ממשו את הפונקציה

`double triangleArea(Triangle *t)`

הפונקציה מחשבת שטח של משולש. הפונקציה מקבלת מצביע למשולש כקלט ומחזירה את שטח המשולש. ניתן להניח כי הקלט תקין (t אינו NULL).

נוסחא לחישוב שטח משולש שקודקודיו הם A, B, C:

$$area = \left| \frac{A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)}{2} \right|$$

כאשר  $A_x$  היא קואורדינטת ה-x של קודקוד A ו- $A_y$  היא קואורדינטת ה-y של קודקוד A. באופן דומה עבור קודקודים B ו-C.



# Solution– lists 2010/a 2 q3-b

---

```
double triangleArea(triangle_t *t)
{
    return fabs((t->a.x * (t->b.y - t->c.y) +
        t->b.x * (t->c.y - t->a.y) +
        t->c.x * (t->a.y - t->b.y)) / 2);
}
```

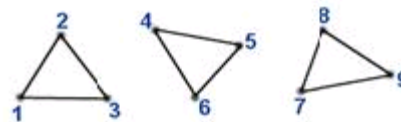
$$area = \left| \frac{A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)}{2} \right|$$

# Example— lists 2010/a 2 q3-c

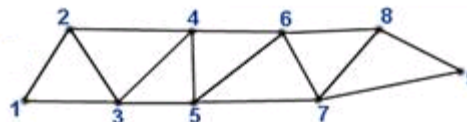
## ■ סעיף ג' (8 נקודות):

■ Mesh הוא מבנה גיאומטרי המכסה תחום במישור בעזרת משולשים. לא תתכן חפיפה כלשהי בין שטחי המשולשים ב mesh אולם ייתכן שיתוף של קודקודים וצלעות.

## ■ איור 1: mesh ללא שיתוף צלעות וקודקודים



## ■ איור 2: mesh עם שיתוף צלעות וקודקודים





# Example– lists 2010/a 2 q3-c

---

- נשמור mesh כרשימה מקושרת של משולשים.
- ממשו את הפונקציה addTriangle המוסיפה משולש ל mesh.  
קלט:  
head - מצביע לראש רשימת המשולשים  
t - מצביע למשולש החדש אותו נרצה להוסיף (ניתן להניח שהמצביע t אינו NULL).  
ערך מוחזר:  
מצביע לראש הרשימה לאחר ההוספה.  
שימו לב: אין משמעות לסדר המשולשים ב- mesh.



## Solution– lists 2010/a 2 q3-c

---

```
triangle_t *addTriangle(triangle_t *mesh, triangle_t *t)
{
    if (t != NULL)
    {
        t->next = mesh;
        return t;
    }
    return mesh;
}
```



# Example– lists 2010/a 2 q3-d

---

■ סעיף ד' (11 נקודות)

■ ממשו פונקציה **רקורסיבית** meshArea המחשבת את שטחו של mesh. הפונקציה מקבלת מצביע לרשימת משולשים ומחזירה את שטח ה-mesh.

```
double meshArea(Triangle *head)
```





## Solution– lists 2010/a 2 q3-d

---

```
double meshArea(triangle_t *mesh)
{
    if (mesh == NULL)
        return 0;
    return triangle_area(mesh) + mesh_area(mesh->next);
}
```



# Solution– lists 2010/a 2 q3

## תכנית המשתמשת בפונקציות הנ"ל

```
int main(){
    triangle_t *mesh = NULL;
    point_t t1_vertices[3] = { {15, 15} , {23, 30} , {50, 25} };
    point_t t2_vertices[3] = { {15, 15} , {45, 10} , {50, 25} };

    triangle_t *t1 = create_triangle(&t1_vertices[0], &t1_vertices[1], &t1_vertices[2]);
    triangle_t *t2 = create_triangle(&t2_vertices[0], &t2_vertices[1], &t2_vertices[2]);

    mesh = add_triangle(mesh, t1);
    mesh = add_triangle(mesh, t2);
    printf("1. triangle area = %g\n", triangle_area(t1));
    printf("2. triangle area = %g\n", triangle_area(t2));
    printf("mesh area = %g\n", mesh_area(mesh));

    free_mesh(mesh);
    return 0;
}
```

הפונקציות  
free\_mesh ו- create\_triangle  
מופיעות בשקפים הבאים



# Solution– lists 2010/a 2 q3

---

```
triangle_t *create_triangle(point_t *a, point_t *b, point_t *c){
    triangle_t *t = NULL;

    if (a == NULL || b == NULL || c == NULL) {
        printf("Error: illegal argument");
        return NULL;
    }

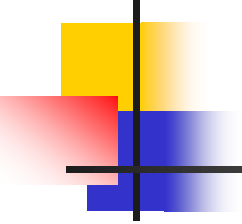
    t = (triangle_t*) malloc(sizeof(triangle_t));
    if (t != NULL){
        t->a = *a;
        t->b = *b;
        t->c = *c;
        t->next = NULL;
    }
    return t;
}
```



# Solution– lists 2010/a 2 q3

---

```
void free_mesh(triangle_t *head)
{
    if (head == NULL)
        return;
    free_mesh(head->next);
    free(head);
}
```



# Example— two dimensional arrays

## 2010/b 1 q3

---

- בשאלה זו תכתבו תכנית המחפשת מילים בתפזורת.
- אנו נייצג את התפזורת באמצעות מערך דו-מימדי בגודל  $100 \times 100$  המכיל תוים.
- הגודל הממשי של התפזורת יכול להיות קטן יותר, וניתן יהיה להסיק אותו מכך שהעמודה והשורה האחרונה בתפזורת יכילו את התו '0' בכל תא.
- ניתן להניח שמספר השורות בתפזורת זהה למספר העמודות.

# Example– two dimensional arrays

## 2010/b 1 q3

לדוגמא, נתונה תפזורת בגודל  $3 \times 3$ , מילים בתפזורת זו:

d	s	a	\0	
k	o	c	\0	
a	n	t	\0	
\0	\0	\0	\0	

100 שורות

100 עמודות

משמאל לימין:

ant an

מלמעלה למטה:

on so son act

אלכסון לכיוון ימינה-למטה:

Dot do

- בשאלה זו, אלו הם שלושת הכיוונים החוקיים של מילים בתפזורת.

- בכל הסעיפים ניתן להניח כי התפזורת המתקבלת הינה במבנה המתואר לעיל, ושכל האותיות בתפזורת ובמילים המתקבלות הן אותיות קטנות בשפה האנגלית.

# Example– two dimensional arrays

## 2010/b 1 q3-a

### ■ סעיף א' (12 נקודות):

- כתבו פונקציה המקבלת מערך דו מימדי המיצג תפזורת, וכן מספרי שורה ועמודה חוקיים בתוך התפזורת.
- בנוסף, מקבלת הפונקציה זוג פרמטרים Hdir ו- Vdir המייצגים כיוון אופקי ואנכי בהתאמה -
  - כאשר Hdir==1 ו- Vdir==0 זהו כיוון אופקי
  - כאשר Hdir==0 ו- Vdir==1 זהו כיוון אנכי
  - כאשר Hdir==1 ו- Vdir==1 זהו אלכסון

[המשך ←](#)

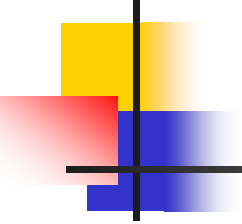
# Example– two dimensional arrays

## 2010/b 1 q3-a

- סעיף א' - המשך:

- הפונקציה תעתיק לתוך המחרוזת `word_copy` את המילה בתפזורת המתחילה בשורה והעמודה הנתונים, וממשיכה בכיוון הנתון ע"י `Hdir` ו-`Vdir`.
- מספר התוים שיועתקו נתון ע"י `num_chars_to_copy`, אלא אם כן לפני כן מגיעים לסוף התפזורת ומופיע התו `'\0'` (ואז יועתקו פחות תוים).
- הניחו שבמחרוזת `word_copy` יש מספיק מקום.





# Solution– two dimensional arrays

## 2010/b 1 q3-a

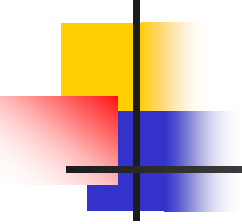
---

```
void copyWord(char crossword[100][100], int row, int col,
int Hdir, int Vdir, char* word_copy,
int num_chars_to_copy) {

    int currRow=row,currCol=col,i=0;

    while (i<num_chars_to_copy && crossword[currRow][currCol]!='\0') {

        word_copy[i]=crossword[currRow][currCol];
        currRow+=Vdir;
        currCol+=Hdir;
        i++;
    }
    word_copy[i]='\0';
}
```



# Example– two dimensional arrays

## 2010/b 1 q3-b

---

- סעיף ב' (12 נקודות):

- כתבו פונקציה המקבלת מערך דו מימדי המיצג תפזורת, וכן מחרוזת נוספת word. הפונקציה בודקת האם המילה word מופיעה בתפזורת באחד משלושת הכיוונים החוקיים. אם כן, הפונקציה תחזיר 1, ואחרת תחזיר את הערך 0.

- ניתן להניח כי אורך המילה word אינו עולה על 30 תווים.

# Solution– two dimensional arrays

## 2010/b 1 q3-b

```
int searchWord(char crossword[100][100], char *word){
    char w[31]; int i,j;
    for (i=0; crossword[i][0]!='\0'; i++) {
        for (j=0; crossword[i][j]!='\0'; j++) {
            copyWord(crossword,i,j,1,0,w,strlen(word));
            if (strcmp(w,word)==0)
                return 1;
            copyWord(crossword,i,j,0,1,w,strlen(word));
            if (strcmp(w,word)==0)
                return 1;
            copyWord(crossword,i,j,1,1,w,strlen(word));
            if (strcmp(w,word)==0)
                return 1;
        }
    }
    return 0;
}
```

# Example– two dimensional arrays

## 2010/b 1 q3-c

### סעיף ג' (10 נקודות):

- כתבו תכנית המחפשת מילים בתפזורת. התכנית תקלוט מילים בלולאה עד להכנסת המילה Stop.
- עבור כל מילה שנקלטה, תבדוק התכנית אם היא קיימת בתפזורת ותודיע הודעה מתאימה. ניתן להניח כי אורך כל מילה אינו עולה על 30 תוים.
- הניחו כי התפזורת נתונה לכם במשתנה ששמו crossword ואין צורך להגדיר אותה. לדוגמא, התפזורת המופיעה בתחילת השאלה תוגדר כך:

```
char crossword[100][100]={{'d','s','a','\0'},  
{'k','o','c','\0'},  
{'a','n','t','\0'},  
{'\0','\0','\0','\0'}};
```

# Example— two dimensional arrays

## 2010/b 1 q3-c

סעיף ג' - דוגמא:

■ דוגמא המתאימה לתפזורת שבתחילת השאלה:

Please enter a word to search: dot

The word dot appears in the crossword

Please enter a word to search : cat

The word cat does not appear in the crossword

d	s	a	\0	
k	o	c	\0	
a	n	t	\0	
\0	\0	\0	\0	

100 עמודות

100 שורות

# Solution– two dimensional arrays

## 2010/b 1 q3-c

```
#include <stdio.h>
#include <string.h>
int main() {
    char word[31];
    int res;
    do {
        printf("Please enter word to search: ");
        scanf("%s",word);
        if (strcmp(word,"Stop")==0)
            break;
        res=searchWord(crossword,word);
        if (res==1)
            printf("word '%s' was found in crossword\n",word);
        else
            printf("word '%s' was NOT found in crossword\n",word);
    } while(1);
}
```



## Example – 2005/a 2 q1-a

---

■ סעיף א' (8 נקודות):

מה עושה הפונקציה הבאה?

אילו הנחות היא מניחה על המצביע שמועבר אליה ועל הטיפוס  
?Data

```
int secret (Data *ptr, double num) {  
    int val;  
    if (ptr == NULL)  
        return 0;  
    val = (ptr->info == num) || secret(ptr->next, num);  
  
    return val;  
}
```



# Solution – 2005/a 2 q1-a

---

## ■ מה הפונקציה עושה?

הפונקציה בודקת אם הערך  $x$  שמועבר אליה נמצא ברשימה. אם כן, מוחזר 1, ואחרת 0.





# Solution – 2005/a 2 q1-a

---

## ■ הנחות של הפונקציה:

- הטיפוס Data הוא מבנה, שמכיל שדה מטיפוס double בשם info ושדה מטיפוס Data\* בשם next.
- המצביע שמועבר אל הפונקציה מצביע על רשימה מקושרת של מבני Data
- האיבר האחרון ברשימה מצביע ל-NULL.



## Example – 2005/a 2 q1-c

---

■ סעיף ג' (8 נקודות):

מה עושה הפונקציה הבאה?

איך ניתן לשנות בה שני תווים בדיוק (לא תווי רווח), כך שלא יחול שינוי בתפקודה?

```
int secret(int a, int b) {  
    if (b < 0)  
        return secret(-a, -b);  
    if (b == 1)  
        return a;  
    else  
        return a + secret(a, b - 1);  
}
```

# Solution – 2005/a 2 q1-c

מה הפונקציה עושה?

הפונקציה מחזירה את הערך של  $a*b$ .

אופציה לשינוי הפונקציה:

```
int secret(int a, int b) {
    if (b < 0)
        return secret(-a, -b);
    old: { if (b == 1)
           return a;
          else
           return a + secret(a, b - 1);
        }
    new: { if (b == 0)
           return 0;
        }
```



## Example – lists 2006/a 2 q3-a

---

■ **סעיף א' (5 נקודות):**

הגדירו **מבנה** (structure) בשם Book, שייצג ספר.

המבנה **יכלול** את שם הספר, שם הסופר, שנת הפרסום, וההכנסות ממכירת הספר.

המבנה יאפשר יצירת **רשימה-מקושרת** של מבני Book.

**הערה:** ניתן להניח שאורך כל אחד מהשמות הוא 80 תווים לכל היותר.



# Solution – lists 2006/a 2 q3-a

---

```
struct _Book {  
    char name[81];  
    char author[81];  
    int year;  
    double income;  
    struct _Book *next;  
}  
typedef struct _Book Book;
```



## Example – lists 2006/a 2 q3-b

---

■ סעיף ב' (10 נקודות):

כתבו פונקציה בעלת המפרט (prototype) הבא:  
`Book *max_book(Book *book_list);`

קלט:

book\_list - מצביע לתחילת רשימה מקושרת של מבני Book.

ערך מוחזר:

מצביע למבנה ברשימה שמייצג את הספר שהכנסות ממנו היו הגבוהות ביותר.



# Solution – lists 2006/a 2 q3-b

---

```
Book *max_book(Book *book_list) {  
    Book *max;  
    max = book_list;  
    while(book_list != NULL) {  
        if (max->income < book_list->income)  
            max=book_list;  
        book_list=book_list->next;  
    }  
    return max;  
}
```



## Example – lists 2006/a 2 q3-c

---

### ■ סעיף ג' (10 נקודות):

כתבו פונקציה בעלת המפרט (prototype) הבא:  
`Book *update_list(Book *book_list, Book *new_book);`

### קלט:

book\_list - מצביע לתחילת רשימה מקושרת של מבני Book  
new\_book - מצביע לספר נוסף

### פעולת הפונקציה וערך מוחזר:

אם הספר הנוסף לא נמצא ברשימה, הפונקציה מוסיפה אותו בסוף הרשימה.  
הפונקציה מחזירה מצביע לרשימה המעודכנת.





# Solution – lists 2006/a 2 q3-c

---

```
Book *update_list(Book *book_list, Book *new_book){
    Book *p_book;

    if (new_book == NULL)
        return book_list;
    if (book_list == NULL)
        return new_book;

    for(p_book = book_list; p_book != NULL; p_book = p_book->next) {
        if(strcmp(p_book->name, new_book->name)==0)
            return book_list;
        if (p_book->next==NULL)
            p_book->next=new_book; /* book_list will be returned */
    }
    /* in the next iteration */
}
```