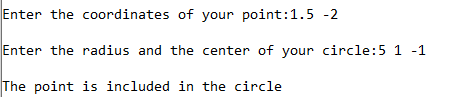
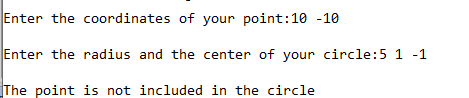
**מעבדה מספר 2 במת"מ –מבנים:**

תרגיל מספר 1:

**מגישים**: **אורי מלכא** 314862996 **ואלן ציפין** 313206062  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
תרגיל מספר 1 פלט 1:   
תרגיל מספר 1 פלט 2:   


#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include<math.h>

typedef struct //struct for point (two coordinates)

{

double x, y;

}point;

typedef struct //struct for circle point and radius

{

point center;

double radius;

}circle;

int is\_incircle(point p, circle c);

/\*

Function name: is\_incircle

Input: point and circle

Output: true - if the point in the circle or on the circle

false - if the point is outside the circle.

\*/

int main()

{

point t;

circle p;

printf("Enter the coordinates of your point:");

scanf("%lf %lf", &t.x, &t.y);

printf("\nEnter the radius and the center of your circle:");

scanf("%lf %lf %lf",&p.radius, &p.center.x, &p.center.y);

is\_incircle(t, p) ? printf("\nThe point is included in the circle") : printf("\nThe point is not included in the circle");

return 0;

}

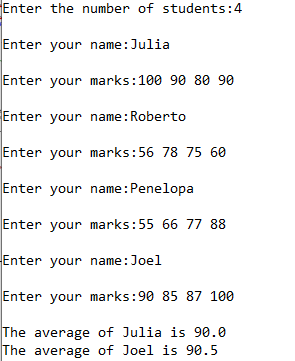
int is\_incircle(point p, circle c) {

double d1 = sqrt(pow((p.x - c.center.x), 2) + pow((p.y - c.center.y), 2)); // d1 is the distance of the point from the center of the circle if its bigger then the radius its means its not within the range.

if (d1 <= c.radius) return 1;

return 0;

}

תרגיל מס' 2:  
  
תרגיל 2 פלט 1:  


#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct stud //structure for student

{

char\* name;

int marks[4];

float avg;

}student;

student\* Create\_Class(int size);

/\*

Function Name: Create\_Class

Input : int (size)

Output : student structure array

Function Algorithm: builds a structure student array in length of "size", receives students name, marks and calls another function to update the average, uses for loops and memo dynamic allocation.

\*/

void Avg\_Mark(student\* s);

/\*

Function Name: Avg\_Mark

Input : Pointer to a student

Output : void

Function Algorithm: Update the student average

\*/

void Print\_One(student\* s);

/\*

Function Name: Print\_One

Input : Pointer to a student

Output : void

Function Algorithm: prints student name and is average

\*/

int main()

{

int size, i;

student\* arr;

printf("\nEnter the number of students:");

scanf("%d", &size);

arr = Create\_Class(size); //receives students structure array

for (i = 0; i < size; i++) { //check if the students avg is greater then 85, if it is prints the name and avg using print\_one function.

if (arr[i].avg > 85)

Print\_One(arr + i);

}

for (i = 0; i < size; i++) //free each name field from the structure array

free(arr[i].name);

free(arr); //free the structure array

return 0;

}

student\* Create\_Class(int size)

{

student\* class;

int i,j;

class = (student\*)malloc(size \* sizeof(student)); //dynamic memo allocation for a student structure array.

if (class == NULL) { //checks if memo allocation failed.

printf("memo allocation failed");

getch();

exit(1); //if failed exits.

}

for (i = 0; i < size; i++)

{

char tempname[50] = ""; // temp name size of 50 , as asked in question description

printf("\nEnter your name:");

scanf("%s", tempname);

class[i].name = (char\*)malloc((strlen(tempname) + 1) \* sizeof(char)); //memo allocate string in the structure name field

if (class[i].name == NULL) {

printf("memo allocation failed"); //check if memo allocation failed

getch();

for (j = 0; j < i; j++) //if failed we need to free all the memo allocated until this point starting from the memo of the strings names in the structure array.

free(class[j].name);

free(class); //free the structure array

exit(1);

}

strcpy(class[i].name, tempname); //using strcpy to put the temp name inside the memo allocated string of the i placed student.

printf("\nEnter your marks:"); //receives the marks from the user.

for (j = 0; j < 4; j++)

scanf("%d", &class[i].marks[j]);

Avg\_Mark(class+i); //update the avg of the certain student.

}

return class; //return the array.

}

void Avg\_Mark(student\* s)

{

int j;

float sum = 0;

for (j = 0; j < 4; j++)

sum += s->marks[j];

s->avg = sum / 4.0; //put the calculated sum inside the certain student avg field.

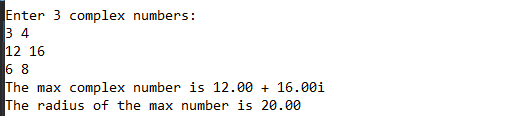
}

void Print\_One(student\* s)

{

printf("\nThe average of %s is %.1f", s->name, s->avg); //prints the name field and the avg field of the certain student structure.

}

תרגיל מס' 3:  
  
  
  
  


#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#define NUM 3

typedef struct //Complex number structure Z=X+YI when r=Real part of Z and i=Imaginary part of Z.

{

double r, i;

}Complex;

double CRadius(Complex z);

/\*

Function name: CRadius

Input: Complex structure

Output: double

Function Algorithm: Calculates the radius of a complex number

\*/

Complex\* CMax(Complex \*a);

/\*

Function name: CMax

Input: Complex structure array

Output: Complex structure

Function Algorithm: returns a pointer to the biggest complex number of the received array.

\*/

int main()

{

Complex arr[NUM], \*max; //max to store pointer that we receive from CMax function

int i;

printf("Enter 3 complex numbers:\n"); //build the complex number array.

for (i = 0; i < NUM; i++)

scanf("%lf %lf", &arr[i].r, &arr[i].i);

max = CMax(arr); //calling CMax function to receive the biggest complex number in the array

printf("The max complex number is %.2lf + %.2lfi\nThe radius of the max number is %.2lf",max->r,max->i, CRadius(\*max));

return 0;

}

double CRadius(Complex z) {

double r = sqrt(pow(z.r, 2) + pow(z.i, 2)); // calcs the radius size of Z.

return r;

}

Complex\* CMax(Complex\* a) {

double arr[NUM] = { 0 }; //help array

int i, maxi=0;

for (i = 0; i < NUM; i++) // we build the help array with the radius size of all the complex numbers, we keep the same indexes of the original structure array.

{

arr[i] = CRadius(a[i]); //call the CRadius function to calculate the size

}

for (i=1;i<NUM;i++)

if (arr[i] > arr[maxi]) maxi = i; //finds the max index

return a + maxi; //returns pointer to the max complex number in the array using the maxi index.

}