TND012/2019: Dugga 2

Attention

This is an **individual** home dugga. Thus, any form of cooperation with other students (e.g. LiTHehack) is seen as academic dishonesty.

Books, notes, and internet material can be used.

If you find any open questions in the presented exercise then feel free to make your own decisions. However, your decisions should be reasonable and must not contradict any of the conditions explicitly written for the exercise. Please, write comments in the program that clarify your assumptions/decisions, if any.

Remember that you should not ignore compiler warnings, since they are usually an indication of serious problems in the code. Remember to set the warning level of your compiler. For Visual Studio, you can follow the instructions given in this file.

Remember also to include in your program all the libraries used in your code (e.g. `#include <cmath>`).

Course goals

This is an assessment of whether you can use

- iteration statements (loops), and
- arrays.

All the points above have been discussed in lectures 1 to 6 and, lessons 1 to 5.

Requirements for grade G

- Readable and well indented code.
- Use of good programming practices.
- Global variables cannot be used.
- Respect for submission instructions.
- Code copied from a web page should be clearly indicated through a commented line containing the web page address used as source, with exception for the course material posted on the course website (e.g. code posted for lectures).
- Use of statements that are not part of the ISO C++ leads to automatically failing the dugga.
- Your programs must pass all test examples shown in this paper.
- Other criteria are listed in the feedback report document available from course website.

Note that there is no guarantee that your code is correct just because it passes all given test examples. Thus, you should always test your code with extra examples, you come up with.

Deadline

30 of September, 12:00.

Submission instructions

1. Submit **one** source code (`.cpp`) file for the exercise through Lisam. The file must be named with your LiU id (e.g. `ninek007.cpp`).
2. The submitted source code file must have your name and personal number at the top of the file.
3. The submitted source code (`.cpp`) file cannot be compressed (e.g. neither `.zip` nor `.rar` files are accepted).

Remember that you should deliver only the source code (`.cpp`) file. Moreover, answers to the dugga exercises sent by email are ignored.

Duggor solutions submitted not accordingly the submission instructions are ignored.

Questions

The aim of the dugga is that you solve the exercise without help of other people. However, we give you the possibility to send us questions about the problem in this dugga by email. If you are a MT student then you should email Aida Nordman (`Aida.Vitoria@liu.se`). If you are an ED or KTS student then you should email Martin Falk (`Martin.Falk@liu.se`).

Only emails received from 17:00 on Friday to 12:00 on Saturday will be answered. Emails received after 12:00 on Saturday are not answered. The emails will be answered until Saturday at 20:00.

Be brief and concrete. Emails can be written either in Swedish or English.

Note that you should not send emails related to Lisam system.

User support for Lisam

Help and support for Lisam system is available at helpdesk@student.liu.se and by calling the phone number 013-28 58 98.

Login and password to access the course material

All course material, like lecture slides and code examples, are available through the course website (**`http://weber.itn.liu.se/~aidvi05/courses/10/index.html`**). However, the material is password protected. Use the following login and password to access the material.

login: **`TND012`** password: **`TND012ht2_12`**

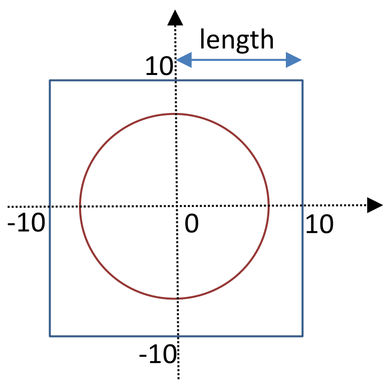
Lycka till!!

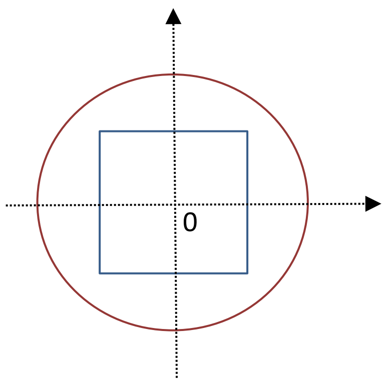
Exercise

Consider a square and a circle both centered at the point $(0,0)$, as the figures below indicate. Write a C++ program that reads a user given list of points in the square and then tests which of those points fall inside the circle. The program must also display the median distance of the points to point $(0,0)$. More concretely, your program should perform the following steps, by the indicated order.

1. Request to the user two integers: the length of half of the square's side (`int`) and the circle's radius (`int`). The length and the radius should be larger or equal to 5. The user input should be validated and if it is not correct (e.g. the user enters -5 for radius) then the program should request the user to re-enter the input data again.
2. Read a list of points. For each point, the user enters two integers, x and y, representing the point's coordinates in a two-dimensional Cartesian coordinate system (this means that the points have integer coordinates). Though the user may enter the same point several times, the program should consider each point only once. In addition, the program should discard any points outside the square. The user marks the end of the list of points with a non-number (e.g. word "`STOP`").
3. Display each point in the list and indicate whether the point falls inside the circle. Recall that a point (x, y) is inside a circle, if $\sqrt{x^2 + y^2}$ is less than the circle's radius.
4. Display the median of the distances from each point to point $(0,0)$[1]. If you consider a sorted list of the distances then the median is the middle value in the sorted list, if the number of values in the list is odd. Otherwise, the median is the average of the two middle values. For instance, the median of the list $1.2\ \ 3.0\ \ 5.0\ \ \mathbf{7.0}\ \ 8.0\ \ 9.6\ \ 11.0$ is 7.0, while the median of $1.2\ \ 3.2\ \ \mathbf{4.0}\ \ \mathbf{5.0}\ \ 7.7\ \ 8.1$ is $(4.0 + 5.0)/2 = 4.5$.

Note that both situations depicted in the example figures below are possible: the circle is totally contained in the square (figure on the left side); the square is inside the circle (figure on the right side).





Assume that the user always enters integer values as input for the program.

Three examples are given below. Values in green color are entered by the user.

Hint: use an array to store the X-coordinates of the points and another array to store the Y-coordinates.

[1] Recall that the program should discard points outside the square and each point is considered only once.

Example 1

```
Radius: 5
Length: 10
Enter the points:
10 -2
3 9
7 6
2 -10
9 10
4 -5
5 22                 ← points outside the square should be read and ignored
3 -4
-3 -2
3 4
3 9                  ← repeated points should be read and ignored
8 8
STOP

(10,-2) NOT in the circle.
(3,9) NOT in the circle.
(7,6) NOT in the circle.
(2,-10) NOT in the circle.
(9,10) NOT in the circle.
(4,-5) NOT in the circle.
(3,-4) NOT in the circle.
(-3,-2) in the circle.
(3,4) NOT in the circle.
(8,8) NOT in the circle.

Median distance to point (0,0): 9.35
```

Example 2

```
Radius: 8
Length: 8
Enter the points:
-1 -3
2 6
-5 8
-8 -7
2 2
0 6
8 7
-5 0
8 5
8 5               ← repeated points should be read and ignored
-2 -7
1 -8
-5 -2
7 -3
-1 -8
7 4
8 -3
6 -8
-7 -8
-1 3
20 -20            ← points outside the square should be read and ignored
0 -3
2 -2
3 2
8 -8
STOP
```

```
(-1,-3) in the circle.
(2,6) in the circle.
(-5,8) NOT in the circle.
(-8,-7) NOT in the circle.
(2,2) in the circle.
(0,6) in the circle.
(8,7) NOT in the circle.
(-5,0) in the circle.
(8,5) NOT in the circle.
(-2,-7) in the circle.
(1,-8) NOT in the circle.
(-5,-2) in the circle.
(7,-3) in the circle.
(-1,-8) NOT in the circle.
(7,4) NOT in the circle.
(8,-3) NOT in the circle.
(6,-8) NOT in the circle.
(-7,-8) NOT in the circle.
(-1,3) in the circle.
(0,-3) in the circle.
(2,-2) in the circle.
(3,2) in the circle.
(8,-8) NOT in the circle.
```

```
Median distance to point (0,0): 7.62
```

Example 3

```
Radius: -12
Length: 6
Not valid input!!

Radius: 12
Length: 4
Not valid input!!

Radius: 12
Length: 6
Enter the points:
22 50                ← points outside the square should be read and ignored
-4 1
-3 0
1 1
6 -2
-6 5
0 0
5 -6
5 -1
-3 4
-1 2
-2 -4
2 -1
-4 4
0 0                  ← repeated points should be read and ignored
-5 3
4 -1
STOP

(-4,1) in the circle.
(-3,0) in the circle.
(1,1) in the circle.
(6,-2) in the circle.
(-6,5) in the circle.
(0,0) in the circle.
(5,-6) in the circle.
(5,-1) in the circle.
(-3,4) in the circle.
(-1,2) in the circle.
(-2,-4) in the circle.
(2,-1) in the circle.
(-4,4) in the circle.
(-5,3) in the circle.
(4,-1) in the circle.

Median distance to point (0,0): 4.47
```