ENGR 151

Fall 2023

Lab 6: Selection, Iteration, and Test-Driven Development

Files to turn in

Please turn in following files on autograder.io

- **fib_for.cpp**
- **fib_while.cpp**
- **park_placement.cpp** - contains your main function that takes in 6 input arguments in the order x1, x2, y1, y2, r1, r2 respectively and prints a single number.

Exercise 1: Fibonacci Sequence

In this exercise, you will practice using `if` statements with conditional operators, `for` loops, and `while` loops.

You will write a program that will print the first `n` numbers of the Fibonacci sequence to the terminal. The number `n` will be requested from the user using `cin`. **You will write 2 versions**: one that uses a `for` loop, and one that uses a `while` loop. Name these **fib_for.cpp** and **fib_while.cpp**, respectively.

You may not use nested loops (loop inside a loop). This is inefficient and not necessary in this case.

Fibonacci numbers are numbers that follow an integer sequence called the Fibonacci sequence. They are characterized by the fact that every number is **the sum of two preceding numbers where the first 2 numbers are 0 and 1.**

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

For example, suppose the program prints “how many fibonacci numbers do you want printed?” and then waits for user input. If the user types “`1`”, then “`0`” should be printed. If the user types “`3`”, then “`0 1 1`” should be printed.

These numbers are named after the Italian Mathematician, **Leonardo of Pisa** known as **Fibonacci** who introduced this sequence in his book, ‘**Liber Abaci**’.

In this book, he considers the growth of an idealized or biologically unrealistic rabbit population, assuming that:

- a pair of rabbits has a pair of children every year
- the children are too young to have children of their own until two years later
- rabbits never die

The puzzle that Fibonacci posed was: how many pairs will there be in 5 years?

- F(1) = 1 -- we start with one pair
- F(2) = 1 -- they're too young to have children the first year
- F(3) = 2 -- in the second year, they have a pair of children
- F(4) = 3 -- in the third year, they have another pair
- F(5) = 5 -- we get the first set of grandchildren

At the end of the n^{th} year, the number of pairs of rabbits is equal to the number of new pairs (which is the number of pairs in the year $n - 2$) plus the number of pairs alive last year($n - 1$). This is the n^{th} Fibonacci number.

No matter if you understand the rabbits or not. All you have to know are the following conditions:

F(0) = 0

F(1) = 1

F(n) = F(n-1) + F(n-2)

Exercise 2: The Park Project Conundrum

In an eco-friendly move, the Gotham city administration decides to construct circular parks all across the city. The city administration issues a request to all interested builders to file their tenders. After receiving an overwhelming response from the builders, the officials decide to go ahead with certain builders. However, they soon realize that some of the park locations described by the builders intersect each other. Before the city administration can finalize the builders, they need your help to figure out whether two circular parks overlap or not.

IMPORTANT NOTE: You should treat these parks as empty circles and not disks (think a jogging path instead of a park).

Assume that Gotham city can be mapped to a 2D Cartesian plane with only positive X and Y axes.

For a point (x,y) in the city

$0 \leq x, y \leq 10^4$

Further, every circular park is described by the coordinates of its center and radius.

For example, a circular park A is determined by its center (5,5) and radius 6 units. In this case, A is a park of 36 π square units with X coordinate of its center as 5 and Y coordinate of its center as 5.

Your program should take as input the location and size of two parks and output their overlap by following the test-driven development approach described above.

Input

Input consists of 6 numbers that describe the location and size of two parks in question in the following manner.

x1 x2 y1 y2 r1 r2

where

`x1` = X coordinate of center of park1

`y1` = Y coordinate of center of park1

`r1` = radius of park1

`x2` = X coordinate of center of park2

`y2` = Y coordinate of center of park2

`r2` = radius of park2

You may assume that inputs for coordinates of the center and radius of the park are **always in the correct range**. (If I didn't tell you this, what kind of test case might you want to add to your test suite?)

Output

Output (cout) a single integer denoting the number of points at which both parks intersect.

Note: In the case of more than 2 points of intersection, output -1. As yourself: why?

Task

For this particular exercise, following the approach of test-driven development,

- First, you should think through all possible test cases. Think about possible outputs and what input conditions would create those outputs. Each case should show 2 circles indicating the location of 2 parks.
- Create a function in your program that takes as inputs the locations of the two parks and returns an integer that denotes the number of intersection points between the two parks. This has been started for you in the file **starter_code_park.cpp** on Canvas. Be sure to rename to **park_placement.cpp** before submitting!

HINTS:

There are at least six possible test scenarios in total.

The Euclidean distance between two points on the plane with Cartesian coordinates (x_1, y_1) and (x_2, y_2) is:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The `cmath` library has `pow`, `sqrt`, and `abs` functions that you may use.