Homework #2

Released: 01-17-2017
Due: 01-24-2017 11:59pm

1 Finding Primes

1.1 Test Primality

Implement a function `is_prime` which uses trial division to test whether the given number p is a prime. If it is, return `true`; otherwise, return `false`.

```
bool is_prime(int p) {
  ...
}
```

1.2 The `main` Function

Please write a program that reads an integer n and prints every prime between 2 and n. You should call `is_prime` in your `main` function.

Input Format

The input has one integer n. We guarantee that $2 \leq n \leq 30000$.

Output Format

Print all the primes between 2 and n, one for each line.

Examples

 # 1

When given the input

```
4
```

Your program should print

```
2
3
```

 # 2

When given the input

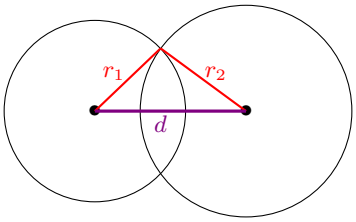
```
17
```

Your program should print

```
2
3
5
7
11
13
17
```

2 Overlapped or Not

In this part, we are going to write a program that determines whether two circles overlap or not. Two circles overlap if the distance between their centers is *less than* the sum of their radii. Tangent circles do not count in this case.



These circles overlap as the purple line is shorter than the total length of the red line.

Every circle is represented using three numbers, giving the coordinate of its center and its radius. These numbers are packed into a struct named `Circle` in our program:

```
struct Circle {
  double x, y;
  double radius;
}
```

2.1 Read Input

Implement a function named `read_circle` that reads a circle and returns it. To be precise, `read_circle` should (1) read three `doubles` x, y and r (2) create a variable of type `Circle` and initialize the variable with x, y and r (3) return that variable.

```
Circle read_circle() {
  ...
}
```

2.2 Test for Overlapping

Implement a function named `overlapped` that returns `true` if the two given circles overlap or `false` otherwise.

```
bool overlapped(Circle C1, Circle C2) {
  ...
}
```

2.3 The `main` Function

Implement the `main` function. Your program will be given one circle C and a sequence of circles $C_1, C_2, \ldots$ to be tested. For each circle C_i in the sequence, please test whether it overlaps with C or not.

Input Format

The first line of the input contains three `doubles` x, y and r, giving the center and the radius of the circle C. We guarantee $r > 0$.

For the following lines, every line also contains three `doubles` x_i, y_i and r_i giving the center and the radius of the circle to be tested. A line with $r_i < 0$ indicates the end of the input. Do not print anything for this line.

Output Format

For every circle being tested, print a line "`overlapped`" if it overlaps with the circle C or "`not overlapped`" otherwise.

Examples

1

When given the input

```
0 0 5
0 2 1
0 10 1
2017 211 -1
```

Your program should print

```
overlapped
not overlapped
```

2

When given the input

```
1 0 1
0 1 0.4
0 1 0.41
0 1 0.414
0 1 0.415
1 -1 0.415
-2017 -211 -2
```

Your program should print

```
not overlapped
not overlapped
not overlapped
overlapped
overlapped
```

3

When given the input

```
1 2 3
-1 -2 -3
```

Your program should print nothing.