Computer Programming A, B, E
FAST-NU, Lahore, Spring 2017

Homework 1

Simulating Josephus Permutation with Dynamic Arrays

Due on Tuesday January 31 11:55 P.M. Marked out of 50 points.

The following is from Wikipedia:

"In computer science and mathematics, the **Josephus problem** (or **Josephus permutation**) is a theoretical problem related to a certain counting-out game.

People are standing in a circle waiting to be executed. Counting begins at a specified point in the circle and proceeds around the circle in a specified direction. After a specified number of people are skipped, the next person is executed. The procedure is repeated with the remaining people, starting with the next person, going in the same direction and skipping the same number of people, until only one person remains, and is freed."

In this assignment we are going to simulate this process on the screen, and we're going to add two small twists to the situation:

i) After each killing the number of people to be skipped this time is chosen at random.
ii) At any point during the process a new bunch of people may be added to those remaining in the circle. This new set of people may be added at any point in the circle but all will go to the same point in a contiguous fashion.

For us, **the direction of skipping is always clock-wise**.

1) What will be displayed on the screen?

At all times we should see the remaining people on the screen in a rectangular arrangement (since it's easier to draw on the console than a circle).

The "current person" is shown in blue (all the other are white).

After each half-second a person is skipped, and the next person becomes blue.

After a person is executed they become red for a half-second before disappearing.

2) What are the persons?

All the "persons" are simply integers: 1, 2…

In the beginning ask the user for an initial number n, and begin with a rectangle of n persons. With person 1 being at the top left of the rectangle.

Most important: **the size of the array in which you're storing the numbers must also be** exactly k. Initially, k=n. After each shooting it goes down by 1. You will have to shrink the array by 1 and remove the shot person from the new array. When new people are added k increases and the array grows.

How will the simulation work?

At the heart of your program is a while loop of the following type:

```
while(true){
//… most of the code will go here
sleep(500);
}
```

Each iteration of this loop takes half-second to complete. Of course, there will also be the time taken to execute the code, but we consider it negligible.

Skeleton Pseudo-code:

At the start of the program do the following:

- Ask the user for value n, and set k=n
- Generate a random number p, between 5 and 8
- Generate a random number r, between 1 and 2k
- **We start counting from a random person between 1 and k.**

In each iteration, the following things will happen:

1) You will move to the next person in the array and turn it blue, unless this is the r^{th} iteration. If this is the r^{th} iteration the current person is shot (turn them red for one iteration.) Update k and shrink the array and copy data without the shot person.
2) In the next iteration a new random value of r is generated, and the process repeats.
3) Adding new people: After each **p** iterations a random number **q,** between 1 and 5, of new people are added to the array, so that the size becomes k=k+q. These people are added starting after a random index **j**, between 0 and k-1. These new people are always numbers greater than all the persons that have been so far. So the first time new people are added they begin from n+1, n+2 and so on

Important: after the first 7 killings no new people are added anymore.

How to draw the rectangle?

Let's say there are k people in the array right now.

To draw them in a rectangle of length l and breadth b, you have solve the following equation:

$$2l + 2b = k$$

This equation has multiple solutions for the values of l and b. But you will pick the values that are closest to each other. For example, if k=46, $l = 11$ and $b = 12$. If k=134, then $l = 33$ $b = 34$ etc.

If k is not an even number, solve the equation for k+1 and add a new person to the circle to adjust the count.

You may have to be careful for smaller values of k.

When does the program stop?

When there is only one person left in the circle, the game stops.

That person (that integer) is declared the winner.

Isn't she lucky?

***

THE END