Computer Programming A, F
FAST-NU, Lahore, Spring 2018

Homework 1

Simulating Josephus Permutation with Dynamic Arrays

Due Thursday February 08 11:55 P.M. Marked out of 50 points.

"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." – Wikipedia

In this assignment, we are going to simulate the Josephus permutation on the console screen, while adding two small twists to the process stated above:

i) After each killing, the number of people to be skipped is changed at random.
ii) At any point during the process a new bunch of people may be added to those that remain in the circle. They will be added at a specified point and will stand contiguously from that point on.

The direction of skipping is always clock-wise.

Further details:

1) What will be displayed on the screen?

The program should display the people on the screen in a rectangular arrangement (since it's easier to draw a rectangle than a circle).

The 'current person' is shown in blue and the rest in white.

After every half a second a person is skipped (turns white) and the next person becomes the 'current person' (turns blue).

After a person is executed they become red for half a second before disappearing, and the rectangle shrinks.

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 and the numbers increasing clockwise.

The numbers in the rectangle are stored in a dynamic array. **The actual physical size of this array is always** 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 (or his or her corpse) from the new array. When m new people are added to the rectangle, k increases by m and the array grows accordingly to accommodate the new people.

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 a second
```

Details of the simulation:

Do the following at the start of the program:

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

Do the following at each iteration:

1) Skip to the next person and turn it blue, unless this is the r^{th} iteration.
2) If this is the r^{th} iteration, shoot the ‘current person’ (turn it red for one iteration), decrement k, shrink the array and generate a new random value r.
3) If this is the p^{th} iteration, generate a random number **m** between 1 and 5 and add m new people to the array, hence increasing its size. Add these new people starting from a random index **j** between 0 and k-1. Represent these new people with numbers greater than all the persons (dead or alive) that have come before. For example, when new people are added for the first time they are given numbers n+1, n+2 … n+m, so on

After the first 10 killings no new people are added to the circle.

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 may solve the following equation:

$$2l + 2b = k$$

This equation has multiple solutions for the values of l and b, but you must 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, then solve the equation for k+1 and add an empty space for the ‘ghost’ person in the rectangle (this person is not there in the array).

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

When does the program stop?

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

That person (that integer) is declared the winner.

Isn’t she lucky?

***

THE END