*Last Updated: 10/13/2024 8:36 PM*

# Assignment: Last Person Alive

*CSC 307 Data Structures and Algorithm Analysis*

**Assignment Objectives**

- Apply an appropriate data structure to solve a programming problem.
- Understand queues by developing a Queue class in C++.

**Tools Used**

1. Installation of Visual Studio Link: https://visualstudio.microsoft.com/downloads/
2. Access to Screenshotting Software (if screenshots are required to complete assignment; Windows Snipping Tool Recommended)
3. Word Processing Document to Create Analysis Report (if analysis questions are asked; Microsoft Office Strongly Recommended)

## Instructions

It is Halloween, and a crazy serial killer has captured you and your friends (and also stole your trick-or-treating candy). The masked killer has you all lined up, side-by-side, and he tells you all that only 1 person will leave this secluded shed alive. He then asks you for a number. You tell him "3". He tells you and your friends that he is going to start pointing at each of you, one by one, starting at the beginning of the line. Once he has counted 3 times, he will kill the person he lands on. Then he will start counting again to the next person, killing each time he counts to three. When he gets to the end, he wraps back around to the front and continues the count. One by one, he kills you and your friends until only one person remains. He takes the candy and leaves you to run for your life.

In this lab, you are going to write a program that can compute the last remaining person based on the rules above. The program should accept a list of all the people (input their names) and then the magic number. The program should then run a simulation of the scenario described above, listing the order in which people are eliminated as well as the surviving member.

To implement this, you will need to use a Queue. You use a queue to simulate this by dequeuing and enqueuing people x number of times until you reach the magic number. Once the number has been reached, you will eliminate the next person in the queue. The general algorithm is as followed:

1. Dequeue from the queue and store the dequeued person in "next."
2. Enqueue next (the person should be added to the back).
3. Repeat steps 1 and 2 (x – 1) number of times.
4. Dequeue the next person and eliminate them. Add the person to the eliminated list.
5. Repeat steps 1 – 4 until the queue has only 1 person remaining.

## Class Design

| class Queue | class Simulation: inherit Queue (as private) |
|---|---|
| #front: Node pointer<br>#back: Node pointer<br>#count: int | -eliminated: string list<br>-survivor: string<br>-skip_number: int |
| +constructor<br>+Enqueue(data: T): none<br>+Dequeue(data: ref T): bool<br>+GetCount(): int | +constructor<br>+RunNewSimulation(string list, integer): void<br>+GetEliminationList(): string list<br>+GetSurvivor(): string<br>-RunSimulation() |

## Description of Methods

Note that the queue methods should reflect standard queue operations. In addition, the lowest access level for all Queue members should be set to protected so that all members are inheritable (for the Simulation class).

Simulation Class:

1. Constructor – Accepts a list of people and the skip number. Calls RunNewSimulation to setup and run the simulation, passing it the constructor parameters.

2. RunNewSimulation – Accepts a list of people and the skip number and sets up the simulation. The eliminated and survivor list should be emptied, and the skip number should be set. The queue should also be cleared. Afterwards, RunSimulation should be called to complete the new simulation.

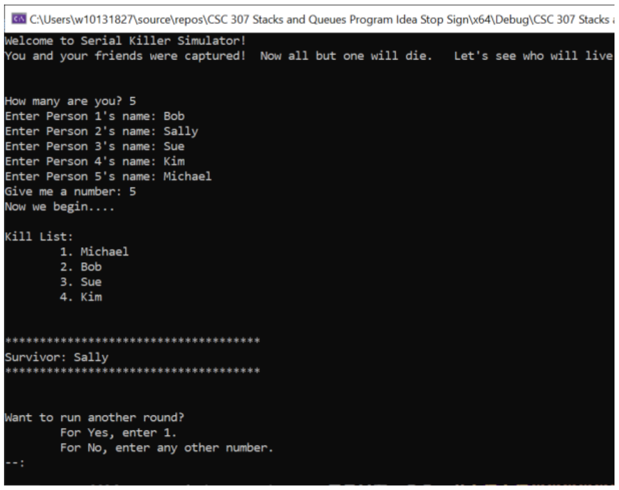
3. GetEliminationList – Returns a vector of the names eliminated in the previous simulation.

4. GetSurvivor – Returns the name of the person that survived the last simulation.

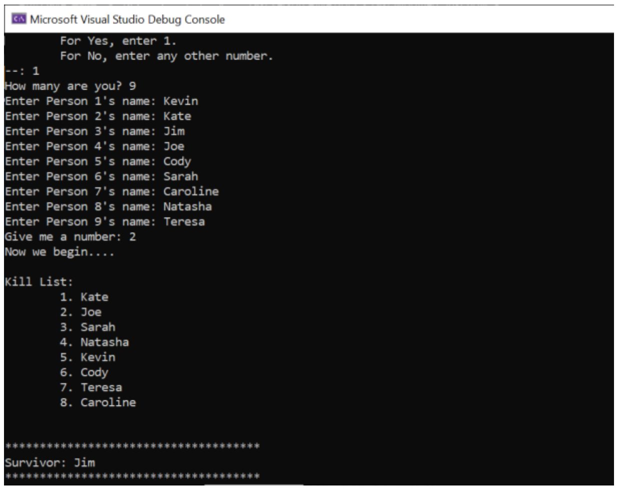
5. RunSimulation – A utility method that supports RunNewSimulation. Conducts the simulation of the currently stored list and skip number.

## Testing

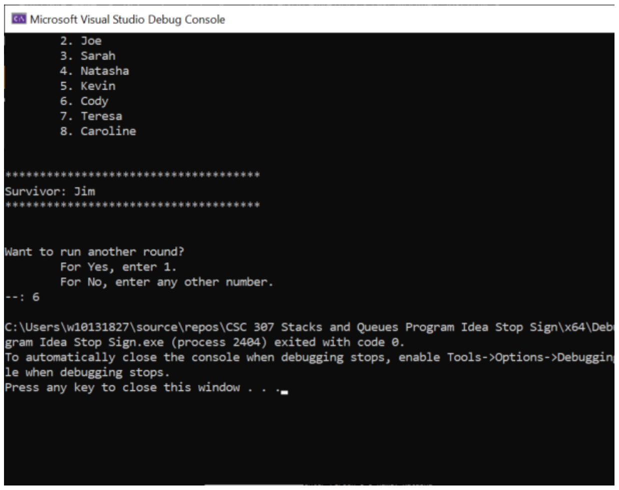
### Sample Output



```
C:\Users\w10131827\source\repos\CSC 307 Stacks and Queues Program Idea Stop Sign\x64\Debug\CSC 307 Stacks a
Welcome to Serial Killer Simulator!
You and your friends were captured!  Now all but one will die.   Let's see who will live


How many are you? 5
Enter Person 1's name: Bob
Enter Person 2's name: Sally
Enter Person 3's name: Sue
Enter Person 4's name: Kim
Enter Person 5's name: Michael
Give me a number: 5
Now we begin....

Kill List:
        1. Michael
        2. Bob
        3. Sue
        4. Kim


*************************************
Survivor: Sally
*************************************


Want to run another round?
        For Yes, enter 1.
        For No, enter any other number.
--:
```



```
Microsoft Visual Studio Debug Console
        For Yes, enter 1.
        For No, enter any other number.
--: 1
How many are you? 9
Enter Person 1's name: Kevin
Enter Person 2's name: Kate
Enter Person 3's name: Jim
Enter Person 4's name: Joe
Enter Person 5's name: Cody
Enter Person 6's name: Sarah
Enter Person 7's name: Caroline
Enter Person 8's name: Natasha
Enter Person 9's name: Teresa
Give me a number: 2
Now we begin....

Kill List:
        1. Kate
        2. Joe
        3. Sarah
        4. Natasha
        5. Kevin
        6. Cody
        7. Teresa
        8. Caroline


*************************************
Survivor: Jim
*************************************
```



```
Microsoft Visual Studio Debug Console
        2. Joe
        3. Sarah
        4. Natasha
        5. Kevin
        6. Cody
        7. Teresa
        8. Caroline


*************************************
Survivor: Jim
*************************************


Want to run another round?
        For Yes, enter 1.
        For No, enter any other number.
--: 6

C:\Users\w10131827\source\repos\CSC 307 Stacks and Queues Program Idea Stop Sign\x64\Deb
gram Idea Stop Sign.exe (process 2404) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debuggin
le when debugging stops.
Press any key to close this window . . .
```

| Rubric | |
|---|---|
| **Total: 100 pts** | |
| ➢ The Queue class is implemented based on the UML diagram above. | 25 pts |
| Implementation of the Simulation Class | 50 pts |
| ➢ The Simulation class is implemented and extends/inherits the Queue class. | 5 pts |
| ➢ The Simulation class correctly implements all the required attributes. | 5 pts |
| ➢ The Simulation class implements RunSimulation correctly. | 15 pts |
| ➢ The Simulation class implements RunNewSimulation correctly. | 10 pts |
| ➢ The Simulation class implements GetEliminationList correctly. | 10 pts |
| ➢ The Simulation class implements GetSurvivor correctly. | 5 pts |
| Implementation of the Main Menu Loop | 25 pts |
| ➢ The main menu is able to run 1 simulation of a user inputted victim list and magic number. | 10 pts |
| ➢ The main menu is able to run as many simulations as the user wishes and quits the program upon user command. | 15 pts |

| Penalties | |
|---|---|
| Late | -10% if less than 24 hours late<br>-30% if less than 48 hours late<br>-100% if more than 48 hours late |
| Assignment Formatting<br>Examples: no header comment, not labeling problem answers, etc. | Up to -20 pts (depending on the issue) |
| Syntactical Errors | -100 pts (automatic zero) |

## Header Comment

**/***

**Name:** ***<your name>***

**NetID:** ***<your Student ID>***

**Program Description:** ***<enter your program description here>***

***/**

## Submission Instructions

This assignment is due by the due date specified in Canvas. You are expected to submit the following files:

1. *firstname_lastname*_Queue.h - File containing your Queue class definition.
2. *firstname_lastname*_Simulation.h - File containing your Simulation class definition.
3. *firstname_lastname*_MainMenu.cpp - File containing the main menu implementation.