# CS 240 - Data Structures and Algorithms I

## Programming Assignment #3

### Due date: 5/15/16 11:59 pm

# Statement

The main aim of this project is to make you conversant with building and implementing queues. You should use the given framework to implement a program that simulates the game of hot potato.

# Details

Implement a program using the provided framework and correctly test your program to ensure that it responds to all cases as expected. You can add extra methods or fields as you see fit, but the framework provided is mandatory. You need to implement the following methods for HotPotato.java, for example

* **public String play(Queue<String> names, int num)**
  + This method should simulate a game of hot potato and return the name of the winner. The parameter names is the queue containing all the players of the game. The parameter num is the number of times the potato will be passed before a person is removed. The num variable can be seen as the number of times a person will be moved to the back before someone is removed. Look at the examples below for clarification.

## Game

To start, there will be a queue of players. The main idea here is the person at the beginning of the queue is the one holding the potato. When he passes the potato to another person, he joins the end of the queue. This counts as one movement. Each person in the front will pass the potato to the person behind them and move to the back until the num amount of movements has been reached. The person remaining at the front of the queue when num movements are reached is removed. Then the game starts again and players will move to the back until num movements have been reached. Again the person at the front at this point will be removed. This cycle happens until only one person remains in the queue. The final person in the queue is the winner and should be returned.

## Note

Remember that the discussion board on BlackBoard is a great place to ask questions regarding topics and get clarification.

# instructions.png

# Examples:

### Example 1: Kitty Rosina Vern Adriana Luba Rona 3

Given Queue: Kitty Rosina Vern Adriana Luba Rona

Given num: 3

**(Queue after each iteration) Action**

1 Rosina Vern Adriana Luba Rona Kitty //Kitty put in back

2 Vern Adriana Luba Rona Kitty Rosina //Rosina put in back

3 Adriana Luba Rona Kitty Rosina Vern //Vern put in back

REMOVED Adriana

1 Rona Kitty Rosina Vern Luba //Luba put in back

2 Kitty Rosina Vern Luba Rona //Rona put in back

3 Rosina Vern Luba Rona Kitty //Kitty put in back

REMOVED Rosina

1 Luba Rona Kitty Vern //Vern put in back

2 Rona Kitty Vern Luba //Luba put in back

3 Kitty Vern Luba Rona //Rona put in back

REMOVED Kitty

1 Luba Rona Vern //Vern put in back

2 Rona Vern Luba //Luba put in back

3 Vern Luba Rona //Rona put in back

REMOVED Vern

1 Rona Luba //Luba put in back

2 Luba Rona //Rona put in back

3 Rona Luba //Luba put in back

REMOVED Rona

Winner = Luba

### Example 2: Debi Joey Verna Clair Bradford Luba 0

In this example the number of iterations given is 0. Since no iterations occur the person at the front is removed until only one remains.

Given Queue: Debi Joey Verna Clair Bradford Luba

Given num: 0

**(Queue after each iteration) Action**

0. Debi Joey Verna Clair Bradford Luba //no person is put in back because num = 0

REMOVED Debi

0. Joey Verna Clair Bradford Luba //no person is put in back because num = 0

REMOVED Joey

0. Verna Clair Bradford Luba //no person is put in back because num = 0

REMOVED Verna

0. Clair Bradford Luba //no person is put in back because num = 0

REMOVED Clair

0. Bradford Luba //no person is put in back because num = 0

REMOVED Bradford

Winner = Luba

### Example 3: Kathey Ella 4

Given Queue: Kathey Ella

Given num: 4

**(Queue after each iteration) Action**

1 Ella Kathey //Kathey is put in back

2 Kathey Ella //Ella is put in back

3 Ella Kathey //Kathey is put in back

4 Kathey Ella //Ella is put in back

REMOVED Kathey //count reaches 4, so Kathey is removed

Winner = Ella

# Sample method invocation

These are examples of the types of parameters that you should be able to pass to your hot potato game.

(Names represent the queue)

Queue<String> names = Forest Robt Vern Verna

play(names, 6) //returns “Robt”

Queue<String> names = Keli Forest

play(names, 16) //returns “Forest”

# Code

Download PA3.zip from the Blackboard under Assignments → Programming Assignments → PA3.

## Code Structure

data/ → This directory contains different TESTCASE data, which will be used for this assignment. test.txt is the input file with different expressions

java/ → This directory contains all the JAVA code. You only need to modify HotPotato.java; Please do not modify other code in this folder. It is mandatory that you use the Queue.java provided.

run.py → A tool to compile, run & test your code.

# Extract and Run

* Download the file PA3.zip. Extract it.
* Copy the directory to your ZFS system by following the basic linux guide provided.
* Let’s say you extracted into ‘PA3’ directory. Now, from the terminal:

$cd PA3

# To run and evaluate your code

$./run.py

# Compatibility

Note that, all assignments will be tested under Linux environment with Python and Oracle Java is installed. Given code might work on other platforms (like Windows, etc.) but has not been tested. Hence, it is encouraged to develop and test your code in a Linux based environment.

# Submission

You should only modify and upload HotPotato.java to Blackboard. Any change in other files will not be accepted and you will not be evaluated in that case.

# Evaluation

There is some held out data set against which your code will be tested and evaluated. Your main aim is to write a HotPotato class following the specifications provided.

# Honor Code

I encourage students to discuss the programming assignments including specific algorithms and data structures required for the assignments. However, students should not share any source code for solution.

Code exists on the web for many problems including some that we may pose in problem sets or assignments. Students are expected to come up with the answers on their own, rather than extracting them from code on the web. This also means that we ask that you do not share your solutions to any of the homework - programming assignments, or problem sets - with any other students. This includes any sort of sharing, whether face-to-face, by email, uploading onto public sites, etc. Doing so will drastically detract from the learning experience of your fellow students.