## National University of Computer and Emerging Sciences, Lahore Campus

| | | | |
|---|---|---|---|
| Course: | Data Structures | Course: | CS 2001 |
| Program: | BS(CS) | Semester: | Fall 2022 |
| Due Date | 23-Oct-2022 at 11:59pm | Total Marks: | 35 |
| Type: | Assignment 3 | Page(s): | 3 |

**Important Instructions:**

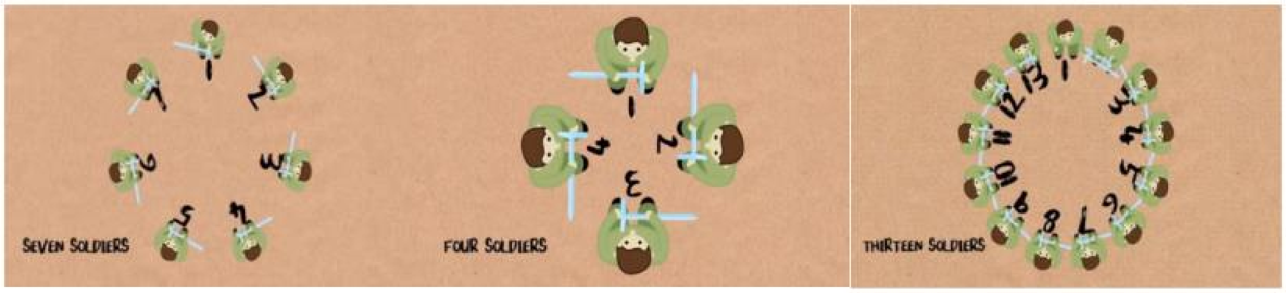
1. Submit your code in a zip file named as your roll number.
2. You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, negative marks will be given to all students involved.
3. Late submission is not allowed.

## Question 1: Josephus Problem

**History:** This problem is named after Flavius Josephus, a Jewish historian living in the 1st century. According to Josephus' account of the siege of Yodfat, he and his 40 soldiers were trapped in a cave by Roman soldiers. They chose suicide over capture, and settled on a serial method of committing suicide by drawing lots. Josephus states that by luck or possibly by the hand of God, he and another man remained until the end and surrendered to the Romans rather than killing themselves. This is the story given in Book 3, Chapter 8, part 7 of Josephus' The Jewish War (writing of himself in the third person)

In computer science and mathematics, the Josephus problem (or Josephus permutation) is a theoretical problem related to a certain counting-out game.
https://en.wikipedia.org/wiki/Josephus_problem

1. People (any number N >1) are standing in a circle waiting to be executed.
2. Counting begins at a specified point (S selected randomly) in the circle and proceeds around the circle in a specified direction. You can assume that it is clock wise
3. After a specified number of people are skipped, say (k-1), the next (kth) 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 (k-1), until only one person remains, and is freed.



Write a C++ program that solves the Josephus problem by using a Queue data structure. Your program should take **N** and **K** input from the user. Design a **graphical user interface** that will update with every move as shown in figure above.

**Note:** This question must be done with a graphical user interface. Otherwise, the this question will not be accepted.

## Question 2: A simple version of snakes and ladders

Implement a game that initially asks how many players will play the game. The game will then ask the name of each player. The game is played as follows:

1. The game is played iteratively. In every iteration, every player takes turn to roll a dice. The face value of dice is added to the total score of the player.
2. The first player to reach 100 score gets 1st position. The second player to reach 100 score gets 2nd position.
3. When a player reaches 100, he no longer remains part of the play. If current score + face value is greater than 100, then the player will retry in the next iteration, i.e., the score must exactly become 100.
4. There are ladders at 20 and 60. If a player's score becomes exactly 20, the score becomes 40. If a player's score is exactly 60, the score becomes 75. Similarly, there are snakes at 50 and 90. If a player's score is exactly 50, the score become 40. If a player's score is exactly 90, the score becomes 79.
5. When n-1 players have reached 100, the game ends and the position of all players that participated in the game is displayed in ascending order.
6. During game play, some useful messages should be displayed, so that one can know what is going on in the game, such as:

```
Please enter the number of players: 2
Please enter the name of Player 1: ABC
Please enter the name of Player 2: XYZ

It is ABC's turn, please enter any key to continue.
ABC got 5. The total score of ABC is now 5.

It is XYZ's turn, please enter any key to continue.
XYZ got 6. The total score of XYZ is now 6.

It is ABC's turn, please enter any key to continue.
ABC got 3. The total score of ABC is now 8.
.
.
.
It is XYZ's turn, please enter any key to continue.
XYZ got 4. The total score of XYZ is 50. There is a snake at 50, so total score is now 40.
```

Use queue for this problem. You can use the queue class available in STL. You can only use push, pop, front, and empty functions of STL queue. Apart from queues, you cannot use any other data structures including arrays in any way (not even for storing the record of players that have already finished the game).