National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Data Structures**	**Course:**	**CS 2001**
	Program:	**BS(CS)**	**Semester:**	**Fall 2023**
	Due Date	**27th October, 2024**	**Total Marks:**	**40**
	Type:	**Assignment 2**	**Instructor:**	**Usama Hassan**

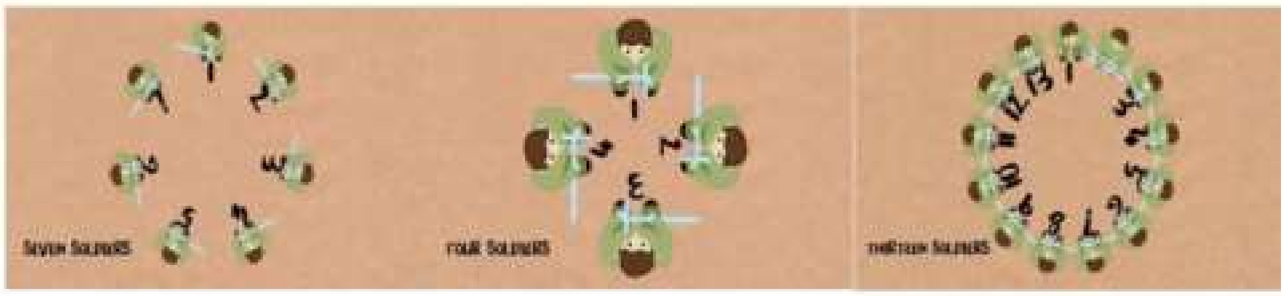
Question 1: Josephus Problem

History: This problem is named after Flavius Josephus, a Jewish historian livinginthe 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 theend and surrendered to the Romans rather than killing themselves. This is the storygiven in Book 3, Chapter 8, part 7 of Josephus' The Jewish War (writing of himself inthe 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 clockwise.



3. After a specified number of people are skipped, say (k-1), the next (kth) personis executed. The procedure is repeated with the remaining people, starting withthenext 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.

For further information on Josephus problem, see The Josephus Problem - Numberphile (but don't use this approach to solve this question)

Question 2: Median in a Binary Search Tree (BST)

Problem Statement:
Use `BST` class that represents a Binary Search Tree (BST) and provides the following methods:

1. **`void insert(T value)`**: Inserts a new value into the BST.
2. **`void delete(T value)`**: Deletes a value from the BST.
3. **`float find_median()`**: Returns the median of all values in the BST. The median is defined as follows:
   - If the number of nodes is odd, the median is the middle element.
   - If the number of nodes is even, the median is the average of the two middle elements.
4. **`vector<T> merge(const BST & other)`**: Merges the two BST, and returns the inorder traversal of the resultant BST in O(N+M) time where N and M are sizes of the two BSTs

Constraints:

- The `insert` and `delete` methods must maintain the properties of the BST.
- The `find_median` method must be efficient and should not require an in-order traversal every time it's called.
- Consider edge cases where the BST is empty or has a single element.

  You may store the size of the BST as a class attribute

Question 3: Minimum Stack

Problem Statement:
Design a class `MinStack` that supports the following operations:

1. **`void push(int value)`**: Pushes the given value onto the stack.
2. **`void pop()`**: Removes the top element from the stack.
3. **`int top()`**: Returns the top element of the stack without removing it.
4. **`get_min()`**: Returns the minimum element in the stack in O(1) time.

Constraints:

- Implement the stack using a single list (or array).
- Ensure that all operations are efficient and do not exceed O(1) time complexity for `get_min()`.
- Ensure that after every update (insert ir delete) having O(1) time complexity, `get_min()` still takes O(1) time.
- Handle cases where the stack may be empty appropriately.

  You may use a temporary stack to maintain minimum element.