Objectives:

In this lab, students will practice:

- Queues
- Recursion

Question 1: Josephus Problem Using Queue

Write a C++ to solve the Josephus problem which has been explained below:

Imagine there are n people standing in a circle, and they are numbered from 1 to n. Starting from a specific person, you count around the circle and eliminate every k-th person. The process continues until only one person remains. The problem is to find the position of the last person standing.

For example, if n = 7 and k = 3, the process would go like this:

- Start at person 1 and count 3 people (1, 2, 3).
- Eliminate person 3.
- Start at person 4 and count 3 people (4, 5, 6).
- Eliminate person 6.
- Start at person 7 and count 3 people (7, 1, 2).
- Eliminate person 2.
- Start at person 4 and count 3 people (4, 5, 7).
- Eliminate person 7.
- Start at person 4 and count 3 people (4, 5, 1).
- Eliminate person 1.
- Now, only person 5 remains, so the position of the last person standing is 5.

Write the time and space complexity as well.(Space complexity means how much memory an algorithm or program uses, including the memory needed for the input data.)

Approach: Using queue data structure, you move the items from one end to another linearly.You are given 'K' the moves we can shift, so after shifting K-1 items to the end of the queue, You can delete the front element.On repeating the above step, a stage will come when only one element will be left, this is the answer itself.

Question 2: Reverse the first K elements of a Queue

You are given a queue of integers and a positive integer K. Your task is to implement a function `reverse_first_k_elements(queue, K)` that reverses the first K elements of the given queue while keeping the rest of the elements in their original order.

reverse_first_k_elements(queue, K):

- You must implement the function `reverse_first_k_elements` using a queue data structure.
- You are not allowed to use any external libraries or built-in functions for reversing.
- Make sure to handle edge cases, such as when K is equal to the length of the queue or when K is 1 or if k > length of queue (reverse the whole queue)

Example:
Input:
queue elements: 1, 2, 3, 4, 5
K = 3
Output: Modified queue: 3, 2, 1, 4, 5

Question 3:
Write a recursive method that for a positive integer returns a string with commas in the appropriate places, for example, putCommas(1234567) returns the string "1,234,567."

Question 4:
Write a recursive method void print01(int k); that prints all 0/1 strings of length k. For example, if k=1, the program should print 0 and 1. If k=2, it should print 00, 01, 10 and 11, etc

Question 5:
Implement a global function stringCompare which compares two-character strings
recursively and:

1. returns 0 if the two strings are equal.
2. If the character of the first string at the index, where the first mismatch occurred, is greater in ASCII value; then it returns 1
3. else it returns -1.

int stringCompare (char const* string1, char const* string2)

Question 6:
Implement a recursive function to find the product of two numbers a and b.
int product(int a, int b)

Question 7:
Imagine you're tasked with creating a C++ program for a financial institution. They want a program that can calculate the future value of a savings account or investment with compound interest, taking into account regular contributions made at the end of each year. Your task is to design a recursive function that can handle this scenario. How would you implement such a recursive function, considering factors like the initial principal, annual interest rate, the number of years, and the regular annual contributions?
Hint :
Calculate interest for one year

```
interest = principal * rate
```

Calculate the new principal after interest and contribution

```
newPrincipal = principal + interest + contribution
```

```
int main()
{
	double principal = 1000.0; // Initial investment
	double rate = 0.05; // Annual interest rate (5%)
	int years = 5; // Number of years
	double contribution = 100.0; // Annual contribution
	double futureValue=calculateFutureValue(principal,rate,years,contribution);
	cout << "The future value of the investment is: $" << futureValue << endl;
	return 0;
}
```