CSE 321 Operating Systems
Lab Assignment 1
Total Marks: 20

Problem 1: Multithreaded Fibonacci Computation and Search [10 Marks]

Problem Statement:

You are tasked with implementing a multithreaded C program that performs the following two tasks using threads:

1. **Fibonacci Sequence Generation**
   The first thread should compute the Fibonacci sequence up to the n-th term (inclusive), where n is an integer input from the user. The result should be returned to the main thread using dynamic memory allocation.

2. **Fibonacci Value Search**
   The second thread should take the computed Fibonacci sequence and search for multiple terms specified by the user. For each search index, it should return the corresponding Fibonacci value if the index is valid, or -1 if the index is invalid.

Sample Input	Sample Output
Enter the term of fibonacci sequence: 8 How many numbers you are willing to search?: 3 Enter search 1: 0 Enter search 2: 4 Enter search 3: 10	a[0] = 0 a[1] = 1 a[2] = 1 a[3] = 2 a[4] = 3 a[5] = 5 a[6] = 8 a[7] = 13 a[8] = 21 result of search #1 = 0 result of search #2 = 3 result of search #3 = -1

Constraints:

- $0 \leq n \leq 40$
- The number of searches s must be greater than 0
- The search indices must be integers

Problem 2: The Sandwich Maker Synchronization Problem [10 Marks]

Problem Statement:

A supplier and three sandwich makers work concurrently to make sandwiches.

- Each **sandwich maker** has an **infinite supply of one unique ingredient**:
    1. Maker A → Bread
    2. Maker B → Cheese
    3. Maker C → Lettuce

To make a sandwich, all **three ingredients** (bread, cheese, lettuce) are needed.

The **supplier** repeatedly places **two random ingredients** on a shared table.
The sandwich maker who has the **third missing ingredient** should:

1. Detect that the other two ingredients are available.
2. Take them from the table.
3. Make and eat the sandwich.
4. Signal the supplier that the table is free.

At any given time, **only one maker** should make a sandwich, and the **supplier must wait** until the table is empty before placing new ingredients.

Goal:

Design a synchronization mechanism that ensures:

- Mutual exclusion on the shared table.
- Correct signaling between the supplier and sandwich makers.
- No busy waiting.
- No deadlocks or starvation.

Input

You will provide the number of times a supplier places ingredients, N as input.

Solution Strategy

- You will have a thread for the supplier, and a thread for each of the sandwich makers.
- Use a **mutex lock** to control access to the shared table.
- Use **semaphores** to signal ingredient availability and to synchronize the supplier with the appropriate sandwich maker.

Sample Input
4
Sample Output
Supplier places: Bread and Cheese Maker C picks up Bread and Cheese Maker C is making the sandwich... Maker C finished making the sandwich and eats it Maker C signals Supplier Supplier places: Cheese and Lettuce Maker A picks up Cheese and Lettuce Maker A is making the sandwich... Maker A finished making the sandwich and eats it Maker A signals Supplier Supplier places: Bread and Lettuce Maker B picks up Bread and Lettuce Maker B is making the sandwich... Maker B finished making the sandwich and eats it Maker B signals Supplier Supplier places: Bread and Cheese Maker C picks up Bread and Cheese Maker C is making the sandwich... Maker C finished making the sandwich and eats it Maker C signals Supplier