C++ Pointers Lab Assignments ‘ABC’ 6-2-2023

1.

Fibonacci strings

The Fibonacci strings are a series of recursivelydefined strings. F_0 is the string **a**, F_1 is the string **bc**, and F_{n+2} is the concatenation of F_n and F_{n+1}. For example, F_2 is **abc**, F_3 is **bcabc**, F_4 is **abcbcabc**, etc. Given a number n and an index k, return the kth character of the string F_n.

2.

Recursive Winner of Circle

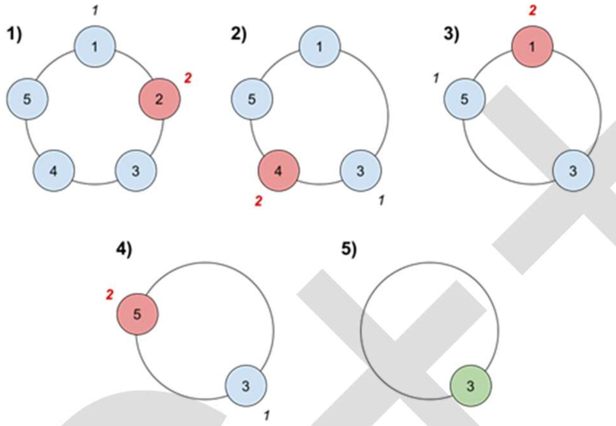
There are n friends that are playing a game. The friends are sitting in a circle and are numbered from 1 to n in **clockwise order**. More formally, moving clockwise from the ith friend brings you to the (i+1)th friend for 1 <= i < n, and moving clockwise from the nth friend brings you to the 1st friend.

The rules of the game are as follows:

1. **Start** at the 1st friend.
2. Count the next k friends in the clockwise direction **including** the friend you started at. The counting wraps around the circle and may count some friends more than once.
3. The last friend you counted leaves the circle and loses the game.
4. If there is still more than one friend in the circle, go back to step 2 **starting** from the friend **immediately clockwise** of the friend who just lost and repeat.
5. Else, the last friend in the circle wins the game.

Given the number of friends, n, and an integer k, code **recursively** to print *the winner of the game*.

Example:



Input: n = 5, k = 2
Output: 3
Explanation: Here are the steps of the game:
1) Start at friend 1.
2) Count 2 friends clockwise, which are friends 1 and 2.
3) Friend 2 leaves the circle. Next start is friend 3.
4) Count 2 friends clockwise, which are friends 3 and 4.
5) Friend 4 leaves the circle. Next start is friend 5.
6) Count 2 friends clockwise, which are friends 5 and 1.
7) Friend 1 leaves the circle. Next start is friend 3.
8) Count 2 friends clockwise, which are friends 3 and 5.
9) Friend 5 leaves the circle. Only friend 3 is left, so they are the winner

Test cases :

Input : 5 2 (n , k) Output : 3 (winner)

Input : 6 5 output: 1

3.	**Combine Row Col Matrix** Read a matrix of M∗N containing lower case English alphabets. Select the first row of the matrix and print all possible combinations as shown below. Code a **recursive function** for the combing. You are allowed to **use only one loop** inside the function **Rules:-** Rule1− Every combination starts from the first row of the matrix and proceeds downwards. You may switch columns though. Rule2− Every combination should have characters equal to the number of rows. Rule3− A combination can't have an element from the same row present twice. *Example*: Input: A[3][2] [a b] [c d] [e f] Output: "ace", "acf", "ade", "adf", "bce", "bcf", "bde", "bdf" (without double quotes).
4.	**Reading, Sorting, and printing of Array of strings using pointers only.** Can use comparestring(str1,str2).
	Education is Not Received **Education is Achieved** **Education is Not a problem** **Education is Notable opportunity** **Education is Not Received** **Education is Practiced** **Intelligent works all ways always ~ KR**