RECURSION - 1 | Week 10

1.PRINT INCREASING DECREASING SEQUENCE.

```
#include <iostream>
using namespace std;

void printIncreasing(int n, int current) {
    if (current > n) return;
    cout << current << " ";
    printIncreasing(n, current + 1);
}

void printDecreasing(int current) {
    if (current < 1) return;
    cout << current << " ";
    printDecreasing(current - 1);
}

void printSequence(int n) {
    printIncreasing(n, 1);
    printDecreasing(n - 1); // To avoid printing 'n' twice
}

int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    printSequence(n);
    return 0;
}
```

2.WAP TO CALCULATE THE SUM OF ODD NUMBERS BETWEEN a and b(include inclusive) BOTH RECURSION.

```
#include <iostream>
using namespace std;

int sumOddNumbers(int a, int b) {
    // Base case: if a is greater than b, stop the recursion
    if (a > b) return 0;

    // Check if the current number 'a' is odd
    if (a % 2 != 0) {
        // If 'a' is odd, add it to the sum and move to the next number
        return a + sumOddNumbers(a + 1, b);
    } else {
        // If 'a' is even, just move to the next number without adding to the sum
        return sumOddNumbers(a + 1, b);
    }
}

int main() {
    int a, b;
    cout << "Enter two numbers (a and b): ";
    cin >> a >> b;

    // Ensure that a is less than or equal to b
    if (a > b) {
        cout << "Invalid input: 'a' should be less than or equal to 'b'." << endl;
        return 1;
    }

    int sum = sumOddNumbers(a, b);
    cout << "Sum of odd numbers between " << a << " and " << b << " is: " << sum << endl;
```

```
    return 0;
}
```

3.GIVEN A POSITIVE INTEGER, RETURN TRUE IF IT IS A POWER OF 2.
INPUT :-64

```
#include <iostream>
using namespace std;

bool isPowerOfTwo(int n) {
    // Base case: 1 is a power of 2 (2^0)
    if (n == 1) return true;
    // Base case: if n is less than 1 or not divisible by 2, it is not a power of 2
    if (n < 1 || n % 2 != 0) return false;
    // Recursive case: divide n by 2 and check if the result is a power of 2
    return isPowerOfTwo(n / 2);
}

int main() {
    int number;
    cout << "Enter a positive integer: ";
    cin >> number;

    if (number <= 0) {
        cout << "Please enter a positive integer." << endl;
        return 1;
    }

    if (isPowerOfTwo(number)) {
        cout << number << " is a power of 2." << endl;
    } else {
        cout << number << " is not a power of 2." << endl;
    }

    return 0;
}
```

4.
CALCULATE NUMBER OF WAYS IN WHICH A PERSON CAN CLIMB N STAIRS IF HE CAN TAKE 1,2 OR EXACTLY 3 STEPS AT EACH LEVEL.

```
#include <iostream>
using namespace std;

int countWays(int n) {
    // Base cases
    if (n == 0) return 1; // There is 1 way to stay at the ground (doing nothing)
    if (n == 1) return 1; // There is 1 way to reach the first step (1 step)
    if (n == 2) return 2; // There are 2 ways to reach the second step (1+1 or 2 steps)
    if (n == 3) return 4; // There are 4 ways to reach the third step (1+1+1, 1+2, 2+1, 3 steps)

    // Recursive case: sum of ways to reach n-1, n-2, and n-3 steps
    return countWays(n - 1) + countWays(n - 2) + countWays(n - 3);
}

int main() {
    int n;
    cout << "Enter the number of stairs: ";
    cin >> n;

    int ways = countWays(n);
    cout << "Number of ways to climb " << n << " stairs is: " << ways << endl;
```

```
    return 0;
}
```