NAME: MOAZZAM FAROOQUI
TASK: DSA ASSIGNMENT#03
INSTRUCTOR: SAMIA MASOOD AWAN
COURSE CODE: CT-159
ROLL NO: CT-24068

Q1. Write a recursive function to compute the factorial of a non-negative integer n.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

int factorial(int n){
    if(n==0 || n==1){
        return 1;
    }
    else{
        return n*factorial(n-1);
    }
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"FACTORIAL OF "<<n<<" IS "<<factorial(n);
    return 0;
}
```

OUTPUT:

```
ENTER A NUMBER:5
FACTORIAL OF 5 IS 120
--------------------------------
Process exited after 3.469 seconds with return value 0
Press any key to continue . . .
```

Q2. Create a recursive function that takes a positive integer and returns the sum of its digits. For example, sum_digits(123) should return 6.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

int sum_of_digits(int n){
    if(n==0){
        return 0;
    }
    else{
        return (n%10)+sum_of_digits(n/10);
    }
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"SUM OF DIGITS IS:"<<sum_of_digits(n)<<endl;
    return 0;
}
```

OUTPUT:

```
ENTER A NUMBER:256
SUM OF DIGITS IS:13

--------------------------------
Process exited after 3.855 seconds with return value 0
Press any key to continue . . .
```

Q3. Write a recursive function to find the nth number in the Fibonacci sequence. The sequence starts with 0, 1 and each subsequent number is the sum of the previous two.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

int fibonacci(int n){
    if(n==0){
        return 0;
    }
    if(n==1){
        return 1;
    }
    else{
        return fibonacci(n-1)+fibonacci(n-2);
    }
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"FIBONACCI SERIES UP TO "<<n<<" TERMS: ";
    for(int i=0;i<n;i++){
        cout<<fibonacci(i)<<" ";
    }
    return 0;
}
```

OUTPUT:

```
ENTER A NUMBER:12
FIBONACCI SERIES UP TO 12 TERMS: 0 1 1 2 3 5 8 13 21 34 55 89
--------------------------------
Process exited after 3.824 seconds with return value 0
Press any key to continue . . .
```

Q4. Write a recursive function to reverse a given string. Do not use any loops or built-in reverse functions.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

void reverse_string(string &str,int start,int end){
    if(start>=end){
        return;
    }
    char temp=str[start];
    str[start]=str[end];
    str[end]=temp;
    reverse_string(str,start+1,end-1);
}

int main(void){
    string str;
    cout<<"ENTER A STRING:";
    cin>>str;
    reverse_string(str,0,str.length()-1);
    cout<<"REVERSED STRING:"<<str;
    return 0;
}
```

OUTPUT:

```
ENTER A STRING:MOAZZAM
REVERSED STRING:MAZZAOM
--------------------------------
Process exited after 6.164 seconds with return value 0
Press any key to continue . . .
```

Q5. Implement a recursive function to check if a given string is a palindrome. A palindrome is a word or phrase that reads the same forwards and backwards.

SOURCE CODE:

```
#include<bits/stdc++.h>
using namespace std;

bool is_palindrome(string&str,int start,int end){
    if(start>=end){
        return true;
    }
    if(str[start]!=str[end]){
        return false;
    }
    return is_palindrome(str,start+1,end-1);
}

int main(void){
    string str;
    cout<<"ENTER A STRING:";
    cin>>str;
    if(is_palindrome(str,0,str.length()-1)){
        cout<<"TRUE!"<<endl;
    }
    else{
        cout<<"FALSE!"<<endl;
    }
    return 0;
}
```

OUTPUT:

```
ENTER A STRING:MADAM
TRUE!

--------------------------------
Process exited after 4.093 seconds with return value 0
Press any key to continue . . .
```

Q6. Write a program to solve the Towers of Hanoi puzzle recursively. The function should take the number of disks and the source, destination, and auxiliary pegs as arguments, and print the steps to move the disks.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

void towerofhanoi(int n,char fromrod,char torod,char helperrod){
    if(n==1){
        cout<<"MOVE DISK "<<n<<" FROM "<<fromrod<<" TO "<<torod<<endl;
        return;
    }

    towerofhanoi(n-1,fromrod,helperrod,torod);

    cout<<"MOVE DISK "<<n<<" FROM "<<fromrod<<" TO "<<torod<<endl;

    towerofhanoi(n-1,helperrod,torod,fromrod);
}

int main(void){
    int n;
    cout<<"ENTER NUMBER OF DISKS:";
    cin>>n;

    towerofhanoi(n,'A','C','B');
    return 0;
}
```

OUTPUT:

```
ENTER NUMBER OF DISKS:3
MOVE DISK 1 FROM A TO C
MOVE DISK 2 FROM A TO B
MOVE DISK 1 FROM C TO B
MOVE DISK 3 FROM A TO C
MOVE DISK 1 FROM B TO A
MOVE DISK 2 FROM B TO C
MOVE DISK 1 FROM A TO C

--------------------------------
Process exited after 2.546 seconds with return value 0
Press any key to continue . . .
```

Q7. Solve the classic N-Queens problem using recursion and backtracking. Place n queens on an n x n chessboard such that no two queens attack each other. Print all possible solutions.

SOURCE CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;

bool isSafe(vector<string>&board,int row,int col,int n){
    for(int j=0;j<n;j++){
        if(board[row][j]=='Q') return false;
    }
    for(int i=0;i<n;i++){
        if(board[i][col]=='Q') return false;
    }
    for(int i=row,j=col;i>=0&&j>=0;i--,j--){
        if(board[i][j]=='Q') return false;
    }
    for(int i=row,j=col;i>=0&&j<n;i--,j++){
        if(board[i][j]=='Q') return false;
    }
    return true;
}
```

```cpp
void nQueens(vector<string>&board,int row,int n,vector<vector<string>>&ans){
    if(row==n){
        ans.push_back(board);
        return;
    }
    for(int j=0;j<n;j++){
        if(isSafe(board,row,j,n)){
            board[row][j]='Q';
            nQueens(board,row+1,n,ans);
            board[row][j]='.';
        }
    }
}
```

```cpp
int main(void){
    int n;
    cout<<"ENTER NUMBER OF QUEENS:";
    cin>>n;

    vector<string>board(n,string(n,'.'));
    vector<vector<string>>ans;
    nQueens(board,0,n,ans);

    cout<<"TOTAL SOLUTIONS:"<<ans.size()<<endl;
    for(int j=0;j<ans.size();j++){
        cout<<"SOLUTION "<<j+1<<":"<<endl;
        for(int i=0;i<n;i++){
            cout<<ans[j][i]<<endl;
        }
        cout<<endl;
    }
    return 0;
}
```

OUTPUT:

```
ENTER NUMBER OF QUEENS:4
TOTAL SOLUTIONS:2
SOLUTION 1:
.Q..
...Q
Q...
..Q.

SOLUTION 2:
..Q.
Q...
...Q
.Q..


--------------------------------
Process exited after 0.9958 seconds with return value 0
Press any key to continue . . .
```

Q8.Categorize and explain the four main types of recursion:

1. Direct Recursion: Where a function calls itself directly.

2. Indirect Recursion: Where a function calls another function, which in turn calls the first function.

3. Tail Recursion: Where the recursive call is the very last operation in the function.

4. Non-Tail Recursion: Where the recursive call is not the last operation, and some computation must
be done with the return value.

For each type, provide a small code snippet or pseudo-code example and explain how it fits the category.

SOURCE CODE OF DIRECT RECURSION:

```cpp
#include<bits/stdc++.h>
using namespace std;

int factorial(int n){
    if(n==0 || n==1){
        return 1;
    }
    else{
        return n*factorial(n-1);//Direct Recursion(Function calls itself directly)
    }
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"FACTORIAL OF "<<n<<" IS "<<factorial(n);
    return 0;
}
```

SOURCE CODE OF INDIRECT RECURSION:

```cpp
#include<bits/stdc++.h>
using namespace std;

void even(int n);
void odd(int n);

void even(int n){
    if(n>0){
        cout<<n<<" IS EVEN"<<endl;
        odd(n-1);//Function calls another function(Indirect recursion)
    }
}

void odd(int n){
    if(n>0){
        cout<<n<<" IS ODD"<<endl;
        even(n-1);//Function calls another function(Indirect recursion)
    }
}

int main(void){
    even(6);
    return 0;
}
```

SOURCE CODE OF TAIL RECURSION:

```cpp
#include<bits/stdc++.h>
using namespace std;

int sum_of_natural_nums(int n,int total){
    if(n==0){
        return total;
    }
    return sum_of_natural_nums(n-1,total+n);//The recursive call is the last operation in the function no further computation
    //happens after this call
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"SUM IS:"<<sum_of_natural_nums(n,0);
    return 0;
}
```

SOURCE CODE OF NON-TAIL RECURSION:

```cpp
#include<bits/stdc++.h>
using namespace std;

int factorial(int n){
    if(n==0 || n==1){
        return 1;
    }
    else{
        return n*factorial(n-1); //Recursive call is NOT the last operation because we multiply n after the call
        //this makes it a non-tail recursion
    }
}

int main(void){
    int n;
    cout<<"ENTER A NUMBER:";
    cin>>n;
    cout<<"FACTORIAL OF "<<n<<" IS "<<factorial(n);
    return 0;
}
```

Q9. Analyze the following recursive functions and identify the type of recursion used in each.
Justify your answer by explaining why it fits a specific category.
• Function A: factorial(n)
• Function B: is_even(n) and is_odd(n)
• Function C: A function to reverse a string where the recursive call is the last step.

Ans: Function A is a Direct and Non tail it is direct because it directly call itself and it does not require any other function to call it and it is a non tail recursion because computation is done as it is multiplied by n so it is a both direct and non tail recursion.

Function B is a Indirect recursion because even calls odd and odd calls even they dont call themselves directly so it is a Indirect recursion.

Function C is a tail recursion because it calls itself directly without any computation is done so we can say that it is a both Direct and Tail recursion.