```cpp
// fibonacci series:
//        0,1,1,2,3,5,8,13
//  index 0,1,2,3,4,5,6,7
//
// fib(n)={
//           0                  n=0
//           1                  n=1
//           fib(n-2)+fib(n-1)  n>1
//         }

// Time Complexity:

// loops- O(n)
// recursion-O(2^n)

#include <iostream>
using namespace std;

int fibloops(int n)
{
    int t0 = 0, t1 = 1, s;
    if (n <= 1)
    {
        return n;
    }
    for (int i = 2; i <= n; i++)
    {
        s = t0 + t1;
        t0 = t1;
        t1 = s;
    }
    return s;
}

int fib(int n)
{
    if (n <= 1)
    {
        return n;
    }
    return fib(n - 2) + fib(n - 1);
}

// Memoization
// reduce the time complexity of the fibonacci series
// create a global or static variable array so that it remains same in the recursive calls and
changes in the variable remains same in the other recursive calls

int F[10];
int fibMem(int n)

{
    if (n <= 1)
    {
        F[n] = n;
        return n;
    }
    else
    {   // if it is -1 i.e it is not called so called the function
```

```
        if (F[n - 2] == -1)
        {
            F[n - 2] = fibMem(n - 2);
        }
        if (F[n - 1] == -1)
        {
            F[n - 1] = fibMem(n - 1);
        }

        return F[n - 2] + F[n - 1];
    }
}
// it calls n+1 times i.e O(n)
int main()
{
    cout<<fib(5)<<endl;
    cout<<fibloops(5)<<endl;
    for (int i = 0; i < 10; i++)
    {
        F[i]=-1;
    }
    cout<<fibMem(5)<<endl;

    return 0;
}
```