Lecture 32

19 January 2022 19:03

RECAP :

Recursive Function

Mandatory :
1. Base Case
2. Recurrence Relation

Optional :
1. Processing

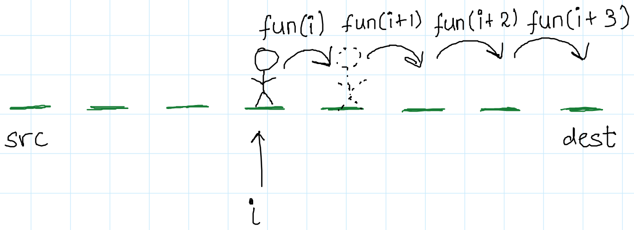
Solving a recursive problem involves `leap of faith' wherein you solve one part / sub-problem and then expect your function to do the rest by calling it recursively.

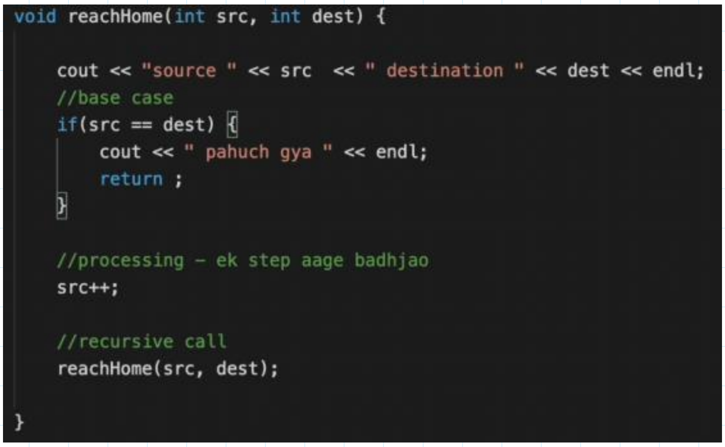
Example : Go from source to destination.

src = 1 , dest = 10. (Use recursion)

Soln : Base Case is when src == dest.

At any given position, I will move one step ahead and then call myself again (Provided src != dest)





```
void reachHome(int src, int dest) {

    cout << "source " << src  << " destination " << dest << endl;
    //base case
    if(src == dest) {
        cout << " pahuch gya " << endl;
        return ;
    }

    //processing - ek step aage badhjao
    src++;

    //recursive call
    reachHome(src, dest);

}
```

Example : Fibonacci Series

0, 1, 1, 2, 3, 5, 8, 13, ...

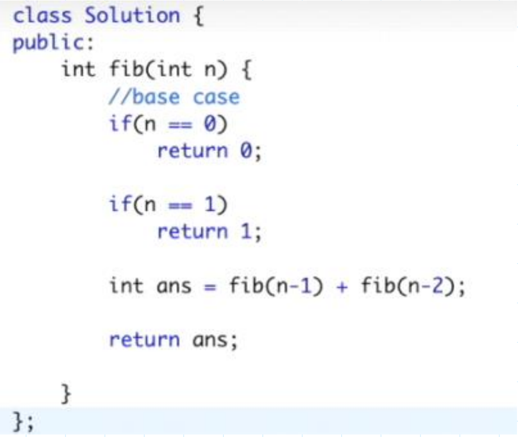
Print n^{th} term of Fibonacci Series.

Soln: $f(n) = f(n-1) + f(n-2)$

But, $f(n) = 0$ where $n <= 1$

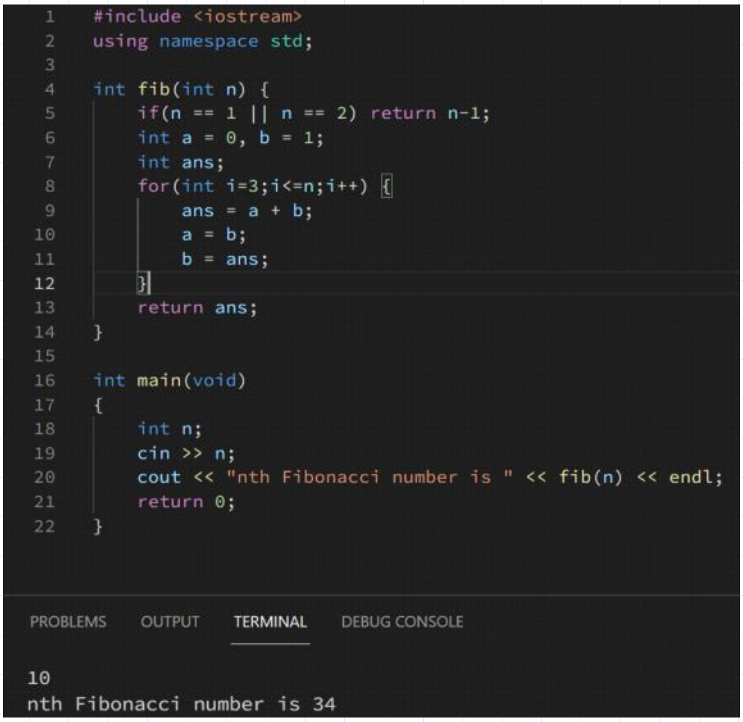
and $f(2) = 1$

Baaki, $f(n) = f(n-1) + f(n-2)$ holds true.



```
class Solution {
public:
    int fib(int n) {
        //base case
        if(n == 0)
            return 0;

        if(n == 1)
            return 1;

        int ans = fib(n-1) + fib(n-2);

        return ans;

    }
};
```

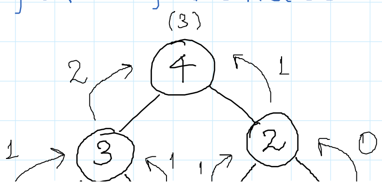
Homework: Solve using for loop.



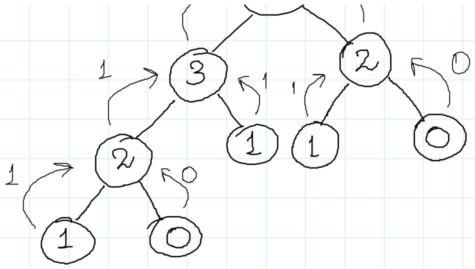
```
#include <iostream>
using namespace std;

int fib(int n) {
    if(n == 1 || n == 2) return n-1;
    int a = 0, b = 1;
    int ans;
    for(int i=3;i<=n;i++) {
        ans = a + b;
        a = b;
        b = ans;
    }
    return ans;
}

int main(void)
{
    int n;
    cin >> n;
    cout << "nth Fibonacci number is " << fib(n) << endl;
    return 0;
}
```

```
10
nth Fibonacci number is 34
```

Recursion Tree for fibonacci :

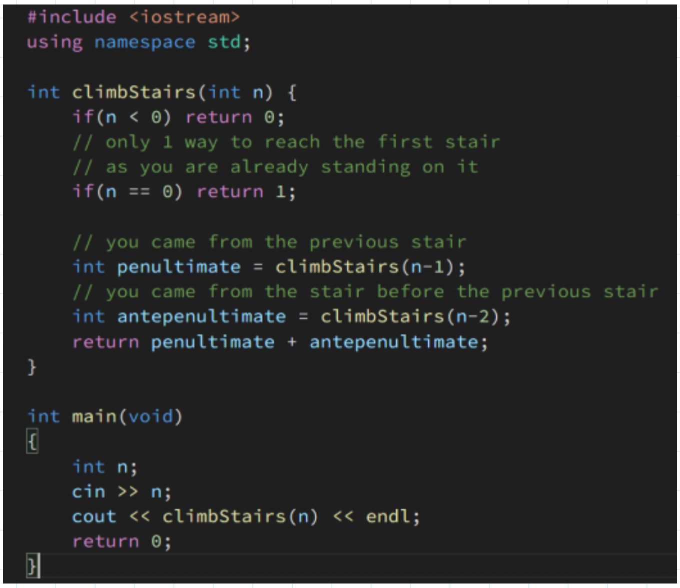


(Zero-based index)



Example : Count ways to reach nth stair.

Soln :



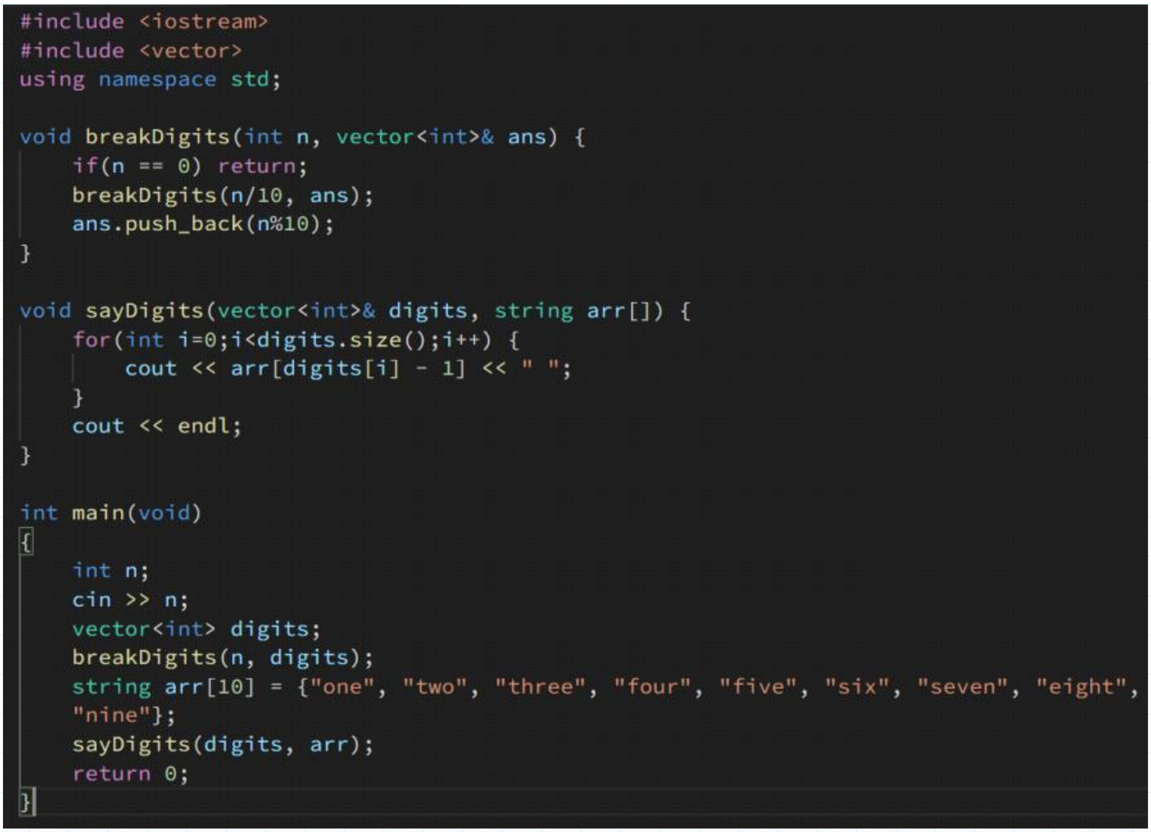
```
#include <iostream>
using namespace std;

int climbStairs(int n) {
    if(n < 0) return 0;
    // only 1 way to reach the first stair
    // as you are already standing on it
    if(n == 0) return 1;

    // you came from the previous stair
    int penultimate = climbStairs(n-1);
    // you came from the stair before the previous stair
    int antepenultimate = climbStairs(n-2);
    return penultimate + antepenultimate;
}

int main(void)
{
    int n;
    cin >> n;
    cout << climbStairs(n) << endl;
    return 0;
}
```

Recursion tree is similar to Fibonacci.

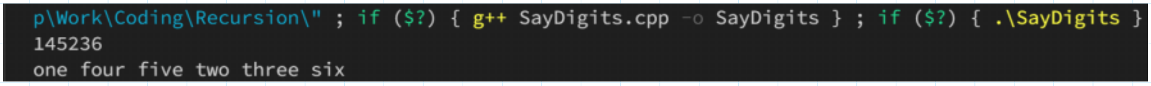
Example : Say digits

I/P - 412

O/P - "four" "one" "two"

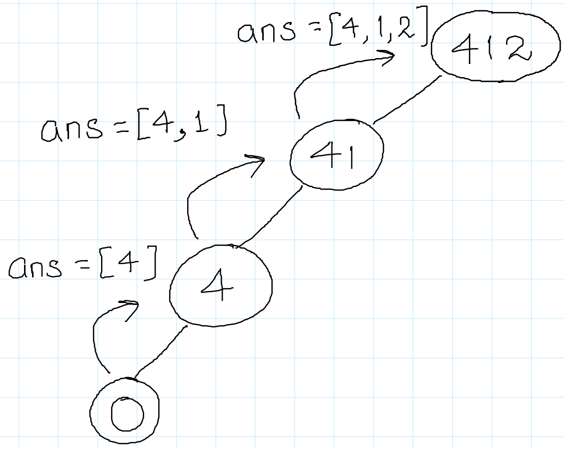


```cpp
#include <iostream>
#include <vector>
using namespace std;

void breakDigits(int n, vector<int>& ans) {
    if(n == 0) return;
    breakDigits(n/10, ans);
    ans.push_back(n%10);
}

void sayDigits(vector<int>& digits, string arr[]) {
    for(int i=0;i<digits.size();i++) {
        cout << arr[digits[i] - 1] << " ";
    }
    cout << endl;
}

int main(void)
{
    int n;
    cin >> n;
    vector<int> digits;
    breakDigits(n, digits);
    string arr[10] = {"one", "two", "three", "four", "five", "six", "seven", "eight",
    "nine"};
    sayDigits(digits, arr);
    return 0;
}
```

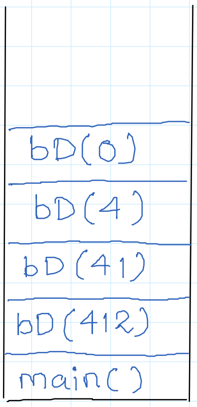


```
p\Work\Coding\Recursion\" ; if ($?) { g++ SayDigits.cpp -o SayDigits } ; if ($?) { .\SayDigits }
145236
one four five two three six
```

Recursion Tree : (Homework)



Function Cell Stack :


bD(0)
bD(4)
bD(41)
bD(412)
main()