RECURSION – 1 SHOT

1. Counting problem

#include<bits/stdc++.h>

using namespace std;

void getCounting(int n) {

//Base Condition

if(n<=0)

return ;

//subproblem or recursive call

getCounting(n-1);

cout<<n<<endl;

//Think, How to print the ascending counting ??

}

int main() {

int n;

cout<<"Please Enter the Input"<<endl;

cin>>n;

cout<<"Counting: "<<endl ;

getCounting(n);

}

1. Factorial:

#include<bits/stdc++.h>

using namespace std;

int getFactorial(int n) {

//Base Condition

if(n<=1)

return 1;

//subproblem or recursive call

int aage\_ka\_factorial = getFactorial(n-1);

//final answer [ye bs 12 tak k liye hi kaam karega]

//HomeWork -> How to find factorial of large numbers.

int answer = n \* aage\_ka\_factorial;

return answer;

}

int main() {

int n;

cout<<"Please Enter the Input"<<endl;

cin>>n;

cout<<"Value of "<<n<<"! is " << getFactorial(n)<<endl;

}

1. Fibonacci Series:

#include <iostream>

using namespace std;

int getFib(int n) {

//Base Condition

if(n==0 || n==1)

return n;

//how to Optimise this, overlapping subproblems ??

return getFib(n-1) + getFib(n-2);

}

int main() {

int n;

cout<<"Please Enter the Input"<<endl;

cin>>n;

//Fib series -> 0,1,1,2,3,5,8,13,......

//0th fibonacci number is 0

//1st fibonacci number is 1

//2nd fibonacci number is 1

//3rd fibonacci number is 2 and so on

cout<<"Value of "<<n<<"th Fibonacci is " << getFib(n)<<endl;

}

1. Print Spelling

#include<bits/stdc++.h>

using namespace std;

//printer for positive integers

void printspell(int n,string str[])

{

    //base case

if(n==0)

return ;

printspell(n/10,str);

    int number = n%10;

    cout<<str[number]<<" ";

cout<<str[n%10]<<" ";

}

int main()

{

int n;

cout<<"Enter the input here:"<<endl;

cin>>n;

string str[10]={"zero","one","two","three","four","five","six","seven","eight","nine"};

cout<<"Answer is :";

printspell(n,str);

return 0;

}

1. Fast Exponentiation:

#include<bits/stdc++.h>

using namespace std;

int getExp(int a, int b) {

if(b==0)

return 1;

//subproblem or recursive call

int aage\_ka\_answer = getExp(a, b-1);

int answer = a \* aage\_ka\_answer;

//is ther any faster way than this ??

return answer;

}

//fast exponentiation

int exp(int n) {

    if(n==0)

    return 1;

//odd

int chotta\_answer = exp(n/2);

    if(n&1) {

        return 2\*chotta\_answer\*chotta\_answer;

    }

    else

    {

        //even

        return chotta\_answer\*chotta\_answer;

    }

}

int expTwo(int n) {

    if(n==0)

    return 1;

    int ans = 2\*expTwo(n-1);

    cout<<" for n "<<n<<" ans "<<ans<<endl;

    return ans;

}

int main() {

//cout<<"Enter the value of 'a' and 'b' "<<endl;

int a,b;

// cin>>a>>b;

cout<<"2 to the power 6 is " << exp(6)<<endl;

// cout<<a<<" to the pwer of "<<b<<" is -> "<<getExp(a,b)<<endl;

}

1. Sorted or Unsorted

#include<bits/stdc++.h>

using namespace std;

//index -> current index of input array

bool checkSorted(vector<int> arr,int index) {

//traversed the entire array

if(index >= arr.size()) {

return true;

}

if(arr[index] < arr[index-1])

return false;

return checkSorted(arr, index+1);

}

int main() {

cout<<"Enter the size of array"<<endl;

int n;

cin>>n;

vector<int> arr(n);

cout<<"Enter Elements: "<<endl;

for(int i=0;i<n;i++) {

cin>>arr[i];

}

bool answer = checkSorted(arr,1);

if(answer)

cout<<"array is sorted "<<endl;

else

cout<<"array is not sorted"<<endl;

}

1. PowerSet:

#include<bits/stdc++.h>

using namespace std;

int totalSubset = 0;

//index-> ith index of input array

// subset: array to store the subset

void printSubset(vector<int> input, vector<int> output, int index)

{

//if saare elements traverse ho chuke hai

if(index>=input.size()) {

//print the output array

for(auto i : output) {

cout<<i<<" ";

}cout<<endl;

        totalSubset++;

return ;

}

//nahi lena hai

printSubset(input,output,index+1);

// lena hai

output.push\_back(input[index]);

printSubset(input,output,index+1);

}

int main() {

cout<<"Enter size"<<endl;

int size;

cin>>size;

vector<int> vec(size);

vector<int> subset; // to store subset, 2^n

cout<<"Enter elements: "<<endl;

for(int i=0; i<size; i++) {

cin>>vec[i];

}

cout<<"Power Set is as follows:"<<endl;

printSubset(vec,subset,0);

cout<<"Total Number of Subsets is "<<totalSubset<<endl;

//should be 2^n

}

1. Jumps – Number of ways to reach destination:

#include<bits/stdc++.h>

using namespace std;

//Problem: https://www.includehelp.com/icp/find-total-ways-to-reach-nth-stair-from-bottom.aspx

int numberOfJumps(int n) {

if(n<0)

return 0;

if(n==0)

return 1;

return numberOfJumps(n-1) + numberOfJumps(n-2) +numberOfJumps(n-3);

}

int main()

{

cout<<"Enter the value of n"<<endl;

int n;

cin>>n;

cout<<"NUMBER OF JUMPS -> "<<numberOfJumps(n)<<endl;

}

1. Subsequence of a string

#include<bits/stdc++.h>

using namespace std;

void getSubsequence(string str, int strIndex, string output) {

//base condition

if(strIndex == str.length()) {

cout<<output<<endl;

return;

}

//nahi lera

getSubsequence(str, strIndex+1, output);

//lera hai

output.push\_back(str[strIndex]);

getSubsequence(str, strIndex+1, output);

}

int main() {

cout<<"Enter the String"<<endl;

string str;

cin>>str;

cout<<"Printing all the Subsequences:"<<endl;

string output="";

getSubsequence(str,0, output);

}

1. Permutation of a String

#include<bits/stdc++.h>

using namespace std;

void getPerm(string str, int index) {

//base condition

if(index>=str.length()){

cout<<str<<endl;

return;

}

for(int i=index;i<str.length();i++) {

swap(str[index],str[i]);

getPerm(str,index+1);

//backtrack

swap(str[index],str[i]);

}

}

int main() {

cout<<"Enter the String"<<endl;

string str;

cin>>str;

cout<<"Printing all the permutations:"<<endl;

getPerm(str,0);

}

1. Source to Destination

#include<bits/stdc++.h>

using namespace std;

map<pair<int,int> ,bool> visited;

//point should be a new point and it should be inside the matrix boundary

bool safeToGo(int a, int b, int m, int n) {

if(a>=0 && a<m && b>=0 && b<n && visited[make\_pair(a,b)]==false) {

return true;

}

return false;

}

void printWays(int m, int n, int src\_x, int src\_y, int dest\_x, int dest\_y, string output) {

visited[make\_pair(src\_x,src\_y)]=true;

//base Condition

if(src\_x==dest\_x && src\_y==dest\_y) {

cout<<"ANSWER -> " <<output<<endl;

visited[make\_pair(src\_x,src\_y)]=false;

return;

}

//RIGHT

if(safeToGo(src\_x +1, src\_y, m,n)) {

output.push\_back('R');

printWays(m,n, src\_x+1, src\_y, dest\_x, dest\_y, output);

output.pop\_back();

}

//UP

if(safeToGo(src\_x , src\_y + 1, m,n)) {

output.push\_back('U');

printWays(m,n,src\_x, src\_y+1 , dest\_x, dest\_y, output);

output.pop\_back();

}

//if we add this line, we get overlapping paths and if we dont add this, we get independent paths

visited[make\_pair(src\_x,src\_y)]=false;

}

int main() {

cout<<"Enter the value of m & n for m\*n matrix"<<endl;

int m,n;

cin>>m>>n;

cout<<"Enter the Origin Co-ordinates"<<endl;

int src\_x, src\_y;

cin>>src\_x>>src\_y;

cout<<"Enter the Destination Co-ordinates"<<endl;

int dest\_x,dest\_y;

cin>>dest\_x>>dest\_y;

string output="";

cout<<"Ways to reach from Origin to Destination are as follows:"<<endl;

printWays(m, n, src\_x, src\_y, dest\_x, dest\_y, output);

}