DAA Lab - Divide and conquer

22MAI1004 - Selvakumar G

Problem: 3D-Fibonacci series is one which is defined as follows:

fiba(0) = 0

fiba(1) = 1

fiba(2) = 2

fiba(n) = fiba(n-1) + fiba(n-2) + fiba(n-3)

Given value of ‘n’, use divide and conquer strategy to find 3D-fiba(n).

Input Forma

First line contains the value of 'n'

Output Format

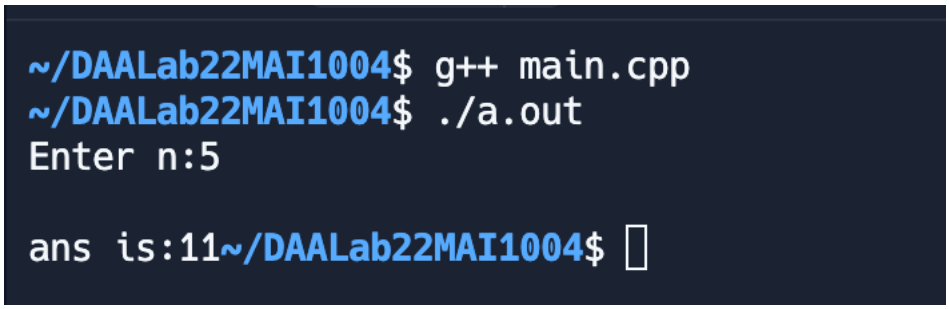
Print the nth term in My Fibonacci series

Code :

```
#include <iostream>
using namespace std;
int fib(int n, int arr[]);
int main()
{
 int n;
 int arr[100];
 cout << "Enter n:";
 cin >> n;
 for(int i=0; i<100; i++)
  {
   arr[i] = -1;
  }
```

```
  cout << "\nans is:" << fib(n, arr);

  return 0;

}

int fib(int n, int arr[])

{

  if(n == 0) return 0;

  else if (n <= 2) return n;

  if (arr[n] != -1) return arr[n];

  arr[n] = fib(n-1, arr) + fib(n-2, arr) + fib(n-3, arr);

  return arr[n];

}
```

Output:



Problem: You are in a shopping mall and saw many giftboxes placed in zig-zag fashion and also occupied much space. You can able to place one giftbox into another if both the width and height of one box are greater than the other box's and the giftboxes are stacked in decreasing value of width and height . Each giftbox represents as Pi=(Wi,Hi) where Wi is the width of the giftbox and Hi is the height of the giftbox. Assume <P1=(W1, H1), P2=(W2, H2), … Pi=(Wi, Hi),…, Pn=(Wn, Hn) > represent 'n' giftboxes , where Wi and Hi are width and height of the giftbox Pi. Let X=(W, H) be a given giftbox. Design an algorithm using Divide-and –Conquer strategy to compute the maximum number of giftboxes are stacked on the top of the given giftbox 'X' to minimize occupied space. Implement your algorithm in any programming language. For example, <P1=(5,4), P2=(6,4), P3=(6,7), P4=(2,3) > and X=(8,10) and maximum number giftboxes are stacked on top of X are 3, they are (6,7), (5,4) and (2,3) .

Code:

```
#include<bits/stdc++.h>
using namespace std;
int check(int arr[100][2],int x,int y,int ind){
	if(ind==-1) return 0;
	if(arr[ind][0]>=x || arr[ind][1]>=y) return check(arr,x,y,ind-1);
	int l = 1 + check(arr,arr[ind][0], arr[ind][1],ind-1);
	int r =  check(arr,x,y,ind-1);
	return max(l,r);
}
int main(){
	int n;
	cout<<"Enter number of boxes \n";
	cin>>n;
	int arr[n][2],x,y;
	cout<<"\n Enter values\n";
	for(int i=0;i<n;i++){
		cin>>arr[i][0]>>arr[i][1];
	}
	cout<<"\n Enter max x and y\n";
	cin>>x>>y;
	for(int i=0;i<n;i++){
		for(int j=i+1;j<n;j++){
			if(arr[i][0]>arr[j][0]|| (arr[i][0]==arr[j][0] && arr[i][1]>arr[j][1])){
				int temp1 = arr[i][0];
				int temp2 = arr[i][1];
				arr[i][0] = arr[j][0];
```

```
                                arr[i][1] = arr[j][1];

                                arr[j][0] = temp1;

                                arr[j][1] = temp2;

                        }

                }

        }

        cout<<"\n ans is ";

        cout<<check(arr,x,y,n-1);

        return 0;

}
```

Output:

```
~/DAALab22MAI1004$ g++ 2.cpp
~/DAALab22MAI1004$ ./a.out
Enter number of boxes
4

 Enter values
5 4
6 4
6 7
7 2

 Enter max x and y
8 10
```

Program: Write a program to find the product of two matrices. You need to find the total number of scalar multiplications and scalar additions are used during the matrix multiplication procedure. Assume the dimensions of the matrices are m X n and n X p.

Input format

Enter the values for m ,n and p

Enter the elements for the first matrix A

Enter the elements for the second matrix B

output format

print elements of AxB

print total number scalar multiplication

print total number of scalar additions

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
	int m,n,p;
	cout<<"Enter m,n,p ";
	cin>>m>>n>>p;
	int mat1[m][n],mat2[n][p],ans[m][p];
	cout<<"Enter matrix 1 ";
	for(int i=0;i<m;i++){
		for(int j=0;j<n;j++){
			cin>>mat1[i][j];
		}
	}
	cout<<"Enter matrix 2 ";
```

```
for(int i=0;i<n;i++){
	for(int j=0;j<p;j++){
		cin>>mat2[i][j];
	}
}
for(int i=0;i<m;i++){
	for(int j=0;j<p;j++){
		ans[i][j] = 0;
	}
}
for(int i=0;i<m;i++){
	for(int j=0;j<p;j++){
		for(int k=0;k<n;k++){
			ans[i][j] += mat1[i][k]*mat2[k][j];
		}
	}
}
for(int i=0;i<m;i++){
	cout<<"\n";
	for(int j=0;j<p;j++){
		cout<<ans[i][j]<<" ";
	}
}
```

```
        cout<<"\nmultiplications and additions are "<<m*n*p<<" , "<<(m*n*p)-(m*p);

}
```

Output: