Nitin Chaudhary
9921103163
F7
APS Lab Week 2

Q1. You are given an array A[m] where first n cells contain integers in sorted order and the rest of the cells are filled with 0. Here assumes m>>n and value of n is unknown. Implement an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in O (log n) time.

Code:

```
// FIRST APPROACH WHICH IS WRONG AS THE ARRAY IS NOT SORTED
#include <iostream>
using namespace std;
int binarySearch(int arr[],int low,int high,int target){
    if (high>=low)
    {
        int mid = (low + high)/2;
        if (arr[mid] == target)
            return mid;
        if (arr[mid] > target)return binarySearch(arr, low, mid-1, target);
        return binarySearch(arr, mid+1, high, target);
    }
    return -1;
}
int main() {
    cout<<"Enter the size of the array\n";
    int n;
    cin>>n;
    int arr[n];
    cout<<"Enter the elements of the array:\n";
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    cout<<"Enter the target to be searched in O(log n) time complexity:";
    int target;
```

```
    cin>>target;
    int index=binarySearch(arr,0,n,target);
    if(index==-1){
        cout<<"The given element is not present in this given array";

    }
    else cout<<"The given element is present at index "<<index;

    return 0;
}
```

```
/tmp/2kJnwQO1Z0.o
Enter the size of the array
6
Enter the elements of the array:
1 2 3 4 5 7
Enter the target to be searched in O(log n) time complexity:7
The given element is present at index 5
```

CORRECT APPROACH WHICH IS TOLD BY SIR IN CLASS-

```
#include <iostream>
using namespace std;
int binarySearch(int arr[],int start,int end,int target){
    if(start<=end){
    int mid=(start+end)/2;
    if(arr[mid]==target)return mid;
    else if(arr[mid]>target)return binarySearch(arr,start,mid-1,target);
    else return binarySearch(arr,mid+1,end,target);
    }
    return -1;
}
int Searchforfirst0(int arr[],int start,int end){
    //cout<<start;
    //cout<<end;
    //if(start<=end){
    int mid=(start+end)/2;
```

```
    if(arr[mid]==0){
        if(arr[mid-1]!=0)return mid;
        else return Searchforfirst0(arr,start,mid-2);
    }
    else{
        return Searchforfirst0(arr,mid+1,end);
    }
    //}
}
int main()
{
    int arr[]={1,2,3,4,5,6,0,0,0,0,0,0,0,0,0,0,0};
    int flag=0;
    cout<<"Enter the value to be searched";
    int n;
    cin>>n;
    if(arr[0]==n){
        //cout<<"hello";
        cout<<"The value is present at index:";
        cout<<0<<endl;
    }
    else
    {
        //cout<<"hello";
    int i=1;
    int index0=1;
    int m=17;
    while(i<m){
        if(arr[i]==0){
            index0=i;
            break;
        }
        else i*=2;
    }
    //cout<<i;
    int first0=Searchforfirst0(arr,i/2,index0);
    //cout<<"first0"<<first0;
    int answer=binarySearch(arr,0,first0,n);
    cout<<"The value is present at index"<<answer;
    return 0;
```

```
  }
}
```

```
Enter the value to be searched4
The value is present at index3

...Program finished with exit code 0
Press ENTER to exit console.
```

Q2. Assume that we are given n pairs of items as input, where the first item is a number and the second item is one of three colours (red, blue, or yellow). Further assume that the items are sorted by number. Give an O(n) algorithm to sort the items by colour (all reds before all blues before all yellows) such that the numbers for identical colours stay sorted. For example: (1, blue), (3, red), (4, blue), (6, yellow), (9, red) should become (3, red), (9, red), (1, blue), (4, blue), (6, yellow).

Code:

```
//WITHOUT USING VECTOR PAIR
#include <bits/stdc++.h>
#include<vector>
using namespace std;
int main() {


    cout<<"Enter the number of pairs you want to enter in the array:";
    int n;
    cin>>n;
    int v1[n];
    string v2[n];
    cout<<"Enter the pairs of the array:\n";
```

```
 for(int i=0;i<n;i++){
    cin>>v1[i]>>v2[i];
    }
int vr1[n]={0};
 string vr2[n];
 int vb1[n]={0};
 string vb2[n];
 int vy1[n]={0};
 string vy2[n];
 int indexr=0;
 int indexb=0;
 int indexy=0;
  for(int i=0;i<n;i++){
     if(v2[i]=="red"){vr1[indexr]=v1[i];
        vr2[indexr]=v2[i];
        indexr++;
     }
     if(v2[i]=="yellow"){vy1[indexy]=v1[i];
        vy2[indexy]=v2[i];
        indexy++;
     }
     if(v2[i]=="blue"){vb1[indexb]=v1[i];
        vb2[indexb]=v2[i];
        indexb++;
     }
  }
  int i=0;
  cout<<"The data after sorting according to requirement is:\n";
  while(vr1[i]!=0){
     cout<<vr1[i]<<" "<<vr2[i]<<endl;
     i++;
  }
  i=0;
  while(vb1[i]!=0){
     cout<<vb1[i]<<" "<<vb2[i]<<endl;
     i++;
  }
  i=0;
  while(vy1[i]!=0){
     cout<<vy1[i]<<" "<<vy2[i]<<endl;
```

```
        i++;
     }
   return 0;
}
```

```
Enter the number of pairs you want to enter in the array:4
Enter the pairs of the array:
1 blue
3 red
6 yellow
9 red
The data after sorting according to requirement is:
3 red
9 red
1 blue
6 yellow


...Program finished with exit code 0
Press ENTER to exit console.
```

```
//USING VECTOR PAIR
#include <iostream>
#include<vector>
using namespace std;
int main() {
    vector< pair <int,string> > v;
    cout<<"Enter the number of pairs you want to enter in the array:";
    int n;
    cin>>n;
    int number;
    string color;
    cout<<"Enter the pairs of the array:\n";
    for(int i=0;i<n;i++){
       cin>>number>>color;
       v.push_back( make_pair(number,color) );
    }
     vector<pair<int,string>> vr,vb,vy;
     for(int i=0;i<v.size();i++){
         if(v[i].second=="red")vr.push_back(make_pair(v[i].first,v[i].second));
         else if(v[i].second=="blue")vb.push_back(make_pair(v[i].first,v[i].second));
```

```
        if(v[i].second=="yellow")vy.push_back(make_pair(v[i].first,v[i].second));
    }
    vector<pair<int,string>> answer;
    for(int i=0;i<vr.size();i++)answer.push_back(make_pair(vr[i].first,vr[i].second));
    for(int i=0;i<vb.size();i++)answer.push_back(make_pair(vb[i].first,vb[i].second));
    for(int i=0;i<vy.size();i++)answer.push_back(make_pair(vy[i].first,vy[i].second));
    cout<<"The approach used to solve this question is that I have created three
buckets for all colors and then I traverse the array and put that pair in it's corresponding
bucket and after that I merge the buckets so that red followed by blue and blue followed
by yellow. as the array is already sorted in number side and hence we will get the
required result";
    cout<<"The array of pairs obtained after performing the required operation in O(n)
time complexity and O (n) space complexity is:\n";
    for(int i=0;i<answer.size();i++)cout<<endl<<answer[i].first<<" "<<answer[i].second;

   return 0;
}
```

```
Enter the number of pairs you want to enter in the array:5
Enter the pairs of the array:
1 blue
3 red
4 blue
6 yellow
9 red
The approach used to solve this question is that I have created three buckets for all
    colors and then I traverse the array and put that pair in it's corresponding bucket
    and after that I merge the buckets so that red followed by blue and blue followed by
    yellow. as the array is already sorted in number side and hence we will get the
    required resultThe array of pairs obtained after performing the required operation
    in O(n) time complexity and O (n) space complexity is:

3 red
9 red
1 blue
4 blue
6 yellow
```

Q3. Find the complexity of the following code snippets:

(a) **Function: One ()**

```
{
        int x; int i; int n;
        x = 20;
        input n;
        for(i = 0; i < n; i++)
                x++;
        output x;
}
```

(b) **Function: Two** (int n)

```
{
        int *x; int i;
        allocate memory for x to
        store n elements
        for(i = 0; i < n; i++)
        {
          input x[i];
          x[i] = x[i] + i;
              output x[i];
      }
}
```

(c) **Function: Three** (int n)

```
{
        int *x; int i;
        allocate memory for x to
        store n elements
        for(i = 0; i < n; i++)

              input x[i];
    for(i = 0; i < n; i++)
              x[i] = x[i] + i;
    for(i = 0; i < n; i++)
              output x[i];
}
```

(d) **Function: Four** (int n, int y)

```
{
        int *x; int i; int j;
        j = 0;
        allocate memory for x to
        store n elements
        for(i = 0; i < n; i++)
              input x[i];
        for(i = 0; i < n; i++)
        {
              if(x[i] == y)
              j++;
    }
    if(j > 0)
        output y is present j times
    else
        output y is not present
}
```

(e) **Function: Five** (int n)

```
{
        int *x; int i; int j, int m; int t;
        allocate memory for x to
        store n elements
        for(i = 0; i < n; i++)
              input x[i];
        for(i = 0; i < n; i++)
        {
              m = x[i];
              t = i;
              for(j = i+1; j < n; j++)
                    {
                  if(m > x[j])
                  {
                      m = x[j];
                      t = j;
                    }
                  }
                  x[t] = x[i];
                  x[i] = m;
        }
        for(i = 0; i < n; i++)
              output x[i];
}
```

(f) **Function: Six** (int m, int n)

```
{
        int **x; int i; int j; int s;
        allocate memory for x to
        store m*n elements
    s = 0;
        for(i = 0; i < n; i++)
        {
          for(j = 0; j < n; j++)
                input x[i][j];
    }
    for(i = 0; i < n; i++)
        {
          for(j = 0; j < n; j++)
                s = s + input x[i][j];
    }
    output s;
}
```

//formulate the required algorithmic time for above function when (a) m = n (b) m ≠ n

a)

(i) ~~for (i=0 ; i<n ; i++) x++;~~
The above statement will get executed for n times

Let $T(n) = n + c = f(n)$

$\Rightarrow f(n) \leq G\, g(n)$ where $g(n) = n$
for $G = c+1$ & $n \geq 1$

$f(n) \leq G\, g(n) \Rightarrow f(n) = O(n)$

&

$G_2\, g(n) \leq f(n)$
for $G_2 = c$

$\Rightarrow f(n) = \Omega(g(n)) = \Omega(n)$

$\Rightarrow G\, g(n) \leq f(n) \leq G_2(g(n))$, $n \geq n_0$

$\Rightarrow f(n) = \theta(g(n))$

$\Rightarrow \boxed{f(n) = \theta(n)}$

b)

(ii) for (i=0 ; i<n ; i++)

This loop will execute for n times

$\Rightarrow f(n) = n + c$

Let $g(n) = n$

$\Rightarrow c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\Rightarrow f(n) = \theta(g(n))$

$\Rightarrow \boxed{f(n) = \theta(n)}$

c)

(c) $T(n) = 3n + c = f(n)$

Let $g(n) = n$

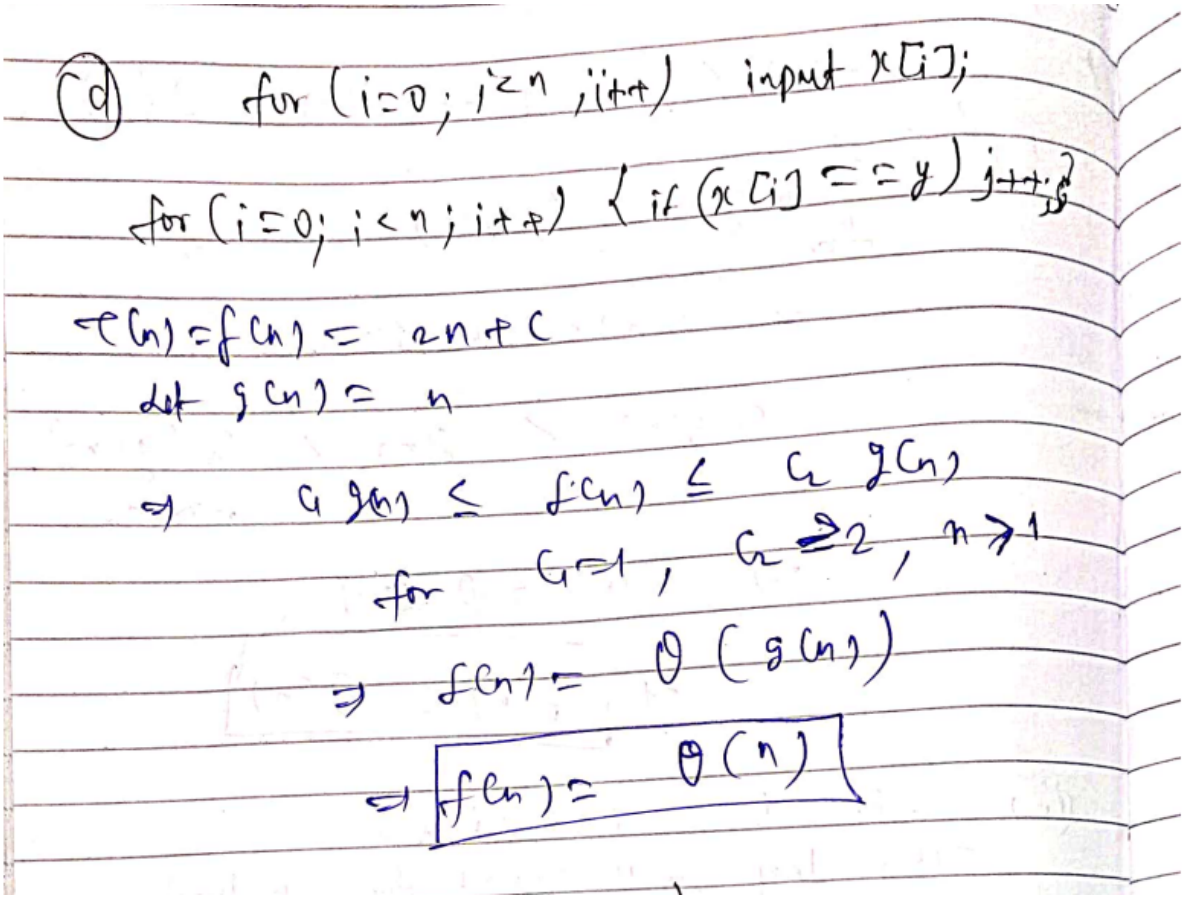
$\Rightarrow c_1 g(n) \leq f(n) \leq c_2 (g(n))$

for $c_1 = 1$, $c_2 \geq 3$, $n \geq 1$

$\Rightarrow f(n) = \theta(g(n))$

$\Rightarrow f(n) = \theta(n)$

d)


(d) for (i=0; i<n ; i++) input x[i];
for (i=0; i<n; i++) { if (x[i] == y) j++; }
T(n) = f(n) = 2n + c
let g(n) = n
⇒ c1 g(n) ≤ f(n) ≤ c2 g(n)
for c1 = 1, c2 ≥ 2, n ≥ 1
⇒ f(n) = Θ(g(n))
⇒ f(n) = Θ(n)

e)

```
(e) for(i=0; i<n; i++)
        input(x[i]);
    for (i=0; i<n; i++)
        m=x[i];
        t=i;
        for (j=i+1; j<n; j++)
        {
            if(m>x[j])
            {
                m=x[j];
                t=j;
            }
        }
    for(i=0; i<n, i++)
```

$T(n) = n + n^2 + n + C$

$T(n) = n^2 + 2n + C$

$\Rightarrow f(n) = n^2 + 2n + C$

$n^2 + 2n \leq cn^2$ for $c \geq 2$

$\Rightarrow O(n^2)$

for $c \leq 1$

$n^2 \leq n^2 + 2n + C$

$\Rightarrow \Omega(n^2)$

$\Rightarrow \boxed{f(n) = \Theta(n^2)}$

f)

(f) T(input) → n^2

T(sum) → $n^2 + c$

$T(n) = 2n^2 + c = f(n)$

Let $g(n) = n^2$

$\Rightarrow f(n) \leq cn^2$ for $c \geq 3$

$\Rightarrow O(n^2)$

$\Leftarrow f(c(g(n)) \leq f(n)$

for $c = 1$

$\Rightarrow \Omega(n^2)$

$\Rightarrow f(n) = \theta(n^2)$

Q4. Implement the recursive algorithms for :
(a) Tower of Hanoi

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
void towerOfHanoi(int n, char from_rod, char to_rod,
                  char aux_rod)
{
    if (n == 0) {
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from rod " << from_rod
         << " to rod " << to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
int main() {
 cout<<"Enter the number of disks:";
 int n;
 cin>>n;
 towerOfHanoi(n,'A','B','C');
 return 0;
}
```

✯ Tower of Hanoi →

$$T(n) = \begin{cases} 2T(n-1)+1, & n \neq 1 \\ 1, & T_1 \end{cases}$$

$$T(n) = 2(2T(n-2)+1)+1$$

$$= 2^2 T(n-2) + 2 + 1$$

$$= 2^3 T(n-3) + 2^2 + 2 + 1$$

$$= 2^n T(n-n) + [2^{n-1} + \ldots + 2^2 + 2 + 1]$$

$T_1 = 1 \Leftarrow n-n=1 \Rightarrow n = n-1$

$$\Rightarrow T(n) = 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots 2^2 + 2 + 1$$

$$\Rightarrow 2T(n) = 2^n + 2^{n-1} + 2^{n-2} + \ldots 2^3 + 2^2 + 2$$

$$\Rightarrow 2(T(n)) - T(n)$$

$$= 2^n - 1$$

$$\Rightarrow \boxed{T(n) = 2^n - 1}$$

$$\Rightarrow \boxed{T(n) = O(2^n)}$$

```
Enter the number of disks:3
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 3 from rod A to rod B
Move disk 1 from rod C to rod A
Move disk 2 from rod C to rod B
Move disk 1 from rod A to rod B
```

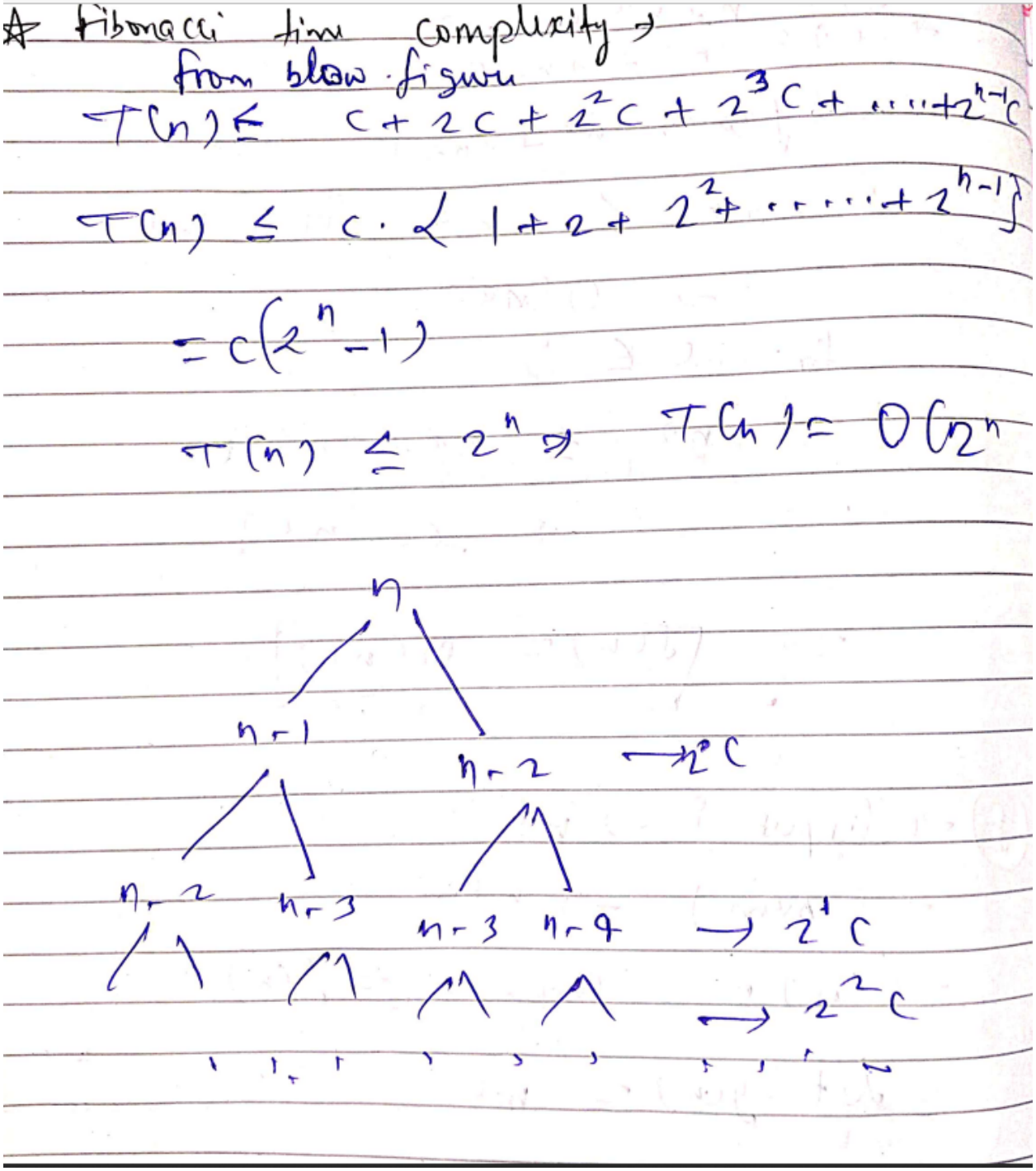
(b) Fibonacci Number computation

Recursive:

```
#include <iostream>
using namespace std;
int fib(int x) {
   if((x==1)||(x==0)) {
      return(x);
   }else {
      return(fib(x-1)+fib(x-2));
   }
}
int main() {
   int x , i=0;
   cout << "Enter the number of terms of series : ";
   cin >> x;
   cout << "\nFibonnaci Series : ";
   while(i < x) {
      cout << " " << fib(i);
      i++;
   }
   return 0;
}
```

Output

```
/tmp/2kJnwQO1Z0.o
Enter the number of terms of series : 7
Fibonnaci Series :  0 1 1 2 3 5 8
```

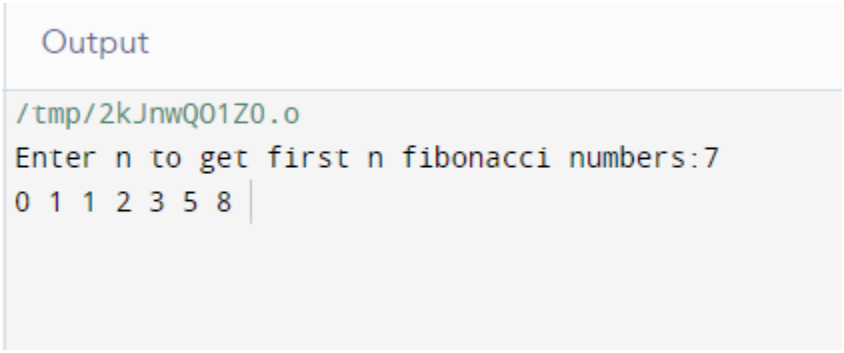


Iterative approach:

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;

int main() {
```

```
 cout<<"Enter n to get first n fibonacci numbers:";
 int n;
 cin>>n;
 int a=0;
 int b=1;
 n-=2;
 cout<<a<<" "<<b<<" ";
 while(n--){
    int c=a+b;
    cout<<c<<" ";
    a=b;
    b=c;
 }
 return 0;
}
```



Q5. Implement the algorithm (Algo_1) presented below and discuss which task this algorithm performs. Also, analyse the time complexity and space complexity of the given algorithm. Further, implement the algorithm with following modification: replace m = ⌈2n/3⌉ with m = ⌊2n/3⌋, and compare the tasks performed by the given algorithm and modified algorithm.
Algo_1(A [0 ... n-1]) { if n = 2 and A[0] > A[1] swap A[0] ↔ A[1] else if n > 2 m = ⌈2n/3⌉ Algo_1 (A [0 .. m − 1]) Algo_1 (A [n – m .. n − 1]) Algo_1 (A [0 .. m − 1]) }

```
Algo_1 ( A [0...n-1] )
{
   if n = 2 and A[0] > A[1]
    swap ( A[0], A[1] )
   else if n > 2
    m = ⌈2n/3⌉

    Algo_1 ( A [0.....m-1] )
    Algo_1 ( A [n-m ... n-1] )
    Algo_1 ( A [0...m-1] )
}
```

for n = 6

n = 4

Algo_1 (A [0, ... 3]) — ①
Algo_1 (A[2 ... 5] — ②
Algo_1 (A [0 ... 3]) — ③

Again → for ① →
n = 2

⇒ Algo_1 (A [0, ... 1])
Algo_1 (A [1, ... 2])
Algo_1 (A [0 ... 1])

Ask to sir in Lab class