DATE: 28-Aug-2020

Himakkshi Jain

Sc: 191112254

DSA: LAB-ASSIGNMENT 3

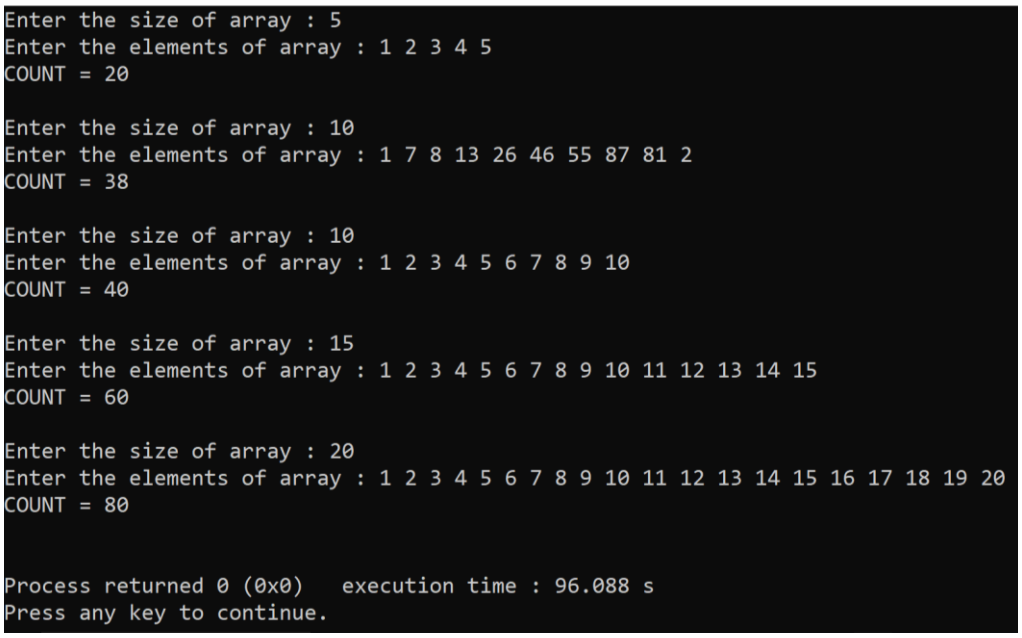
Ques (a): Count number of steps for **SECRET - ALGORITHM.**

Code:

```
#include<iostream>
using namespace std;
int count = 0;

int secret ( int A[] , int n)
{
   int minval = A[0] ;
   ::count++ ;
   int maxval = A[0] ;
   ::count++ ;

   for (int i=1 ; i<n ; i++)
   {
      ::count++ ;
      if ( A[i] < minval )
      {
         minval = A[i] ;
         ::count++ ;
      }
```

```
        ::count++ ;
        if ( A[i] > maxval)
        {
            maxval = A[i] ;
            ::count++ ;
        }

        ::count++ ;
    }

    ::count++ ;   // since the loop condition will be checked n+1 times

    ::count++ ;
    return (maxval - minval);
}

int main()
{
    for (int i=0 ; i<5 ; i++)
    {
        int n ;
        cout << "Enter the size of array : " ;
        cin >> n ;
        int A[n] ;
        cout << "Enter the elements of array : " ;
        for (auto &x : A)
            cin >> x ;

        secret( A , n ) ;
        cout << "COUNT = " << ::count << " \n\n" ;
        ::count = 0 ;
    }
}
```

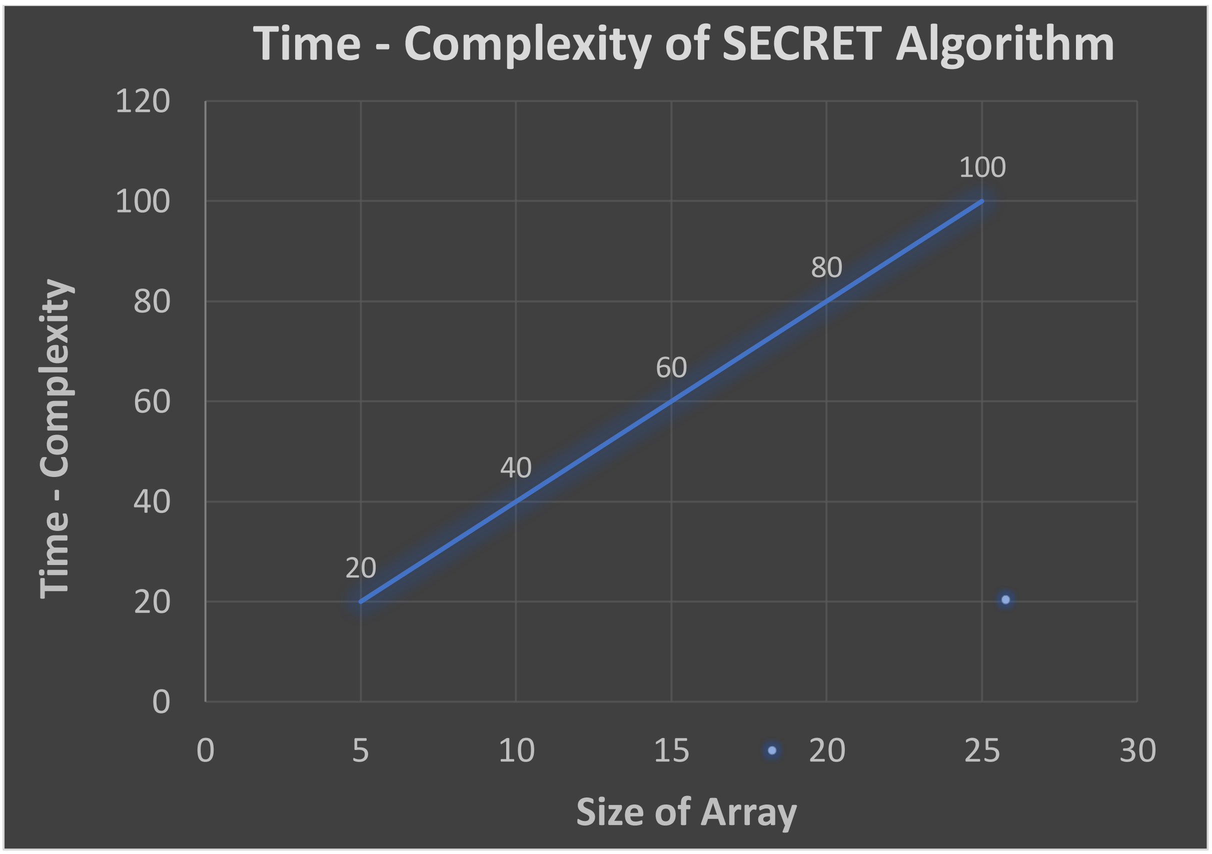
Output:



```
Enter the size of array : 5
Enter the elements of array : 1 2 3 4 5
COUNT = 20

Enter the size of array : 10
Enter the elements of array : 1 7 8 13 26 46 55 87 81 2
COUNT = 38

Enter the size of array : 10
Enter the elements of array : 1 2 3 4 5 6 7 8 9 10
COUNT = 40

Enter the size of array : 15
Enter the elements of array : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
COUNT = 60

Enter the size of array : 20
Enter the elements of array : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
COUNT = 80


Process returned 0 (0x0)   execution time : 96.088 s
Press any key to continue.
```

Graph:

Size of Array	Time - Complexity
5	20
10	40
15	60
20	80
25	100


Time - Complexity of SECRET Algorithm
Time - Complexity
Size of Array
0
20
40
60
80
100
120
0
5
10
15
20
25
30
20
40
60
80
100

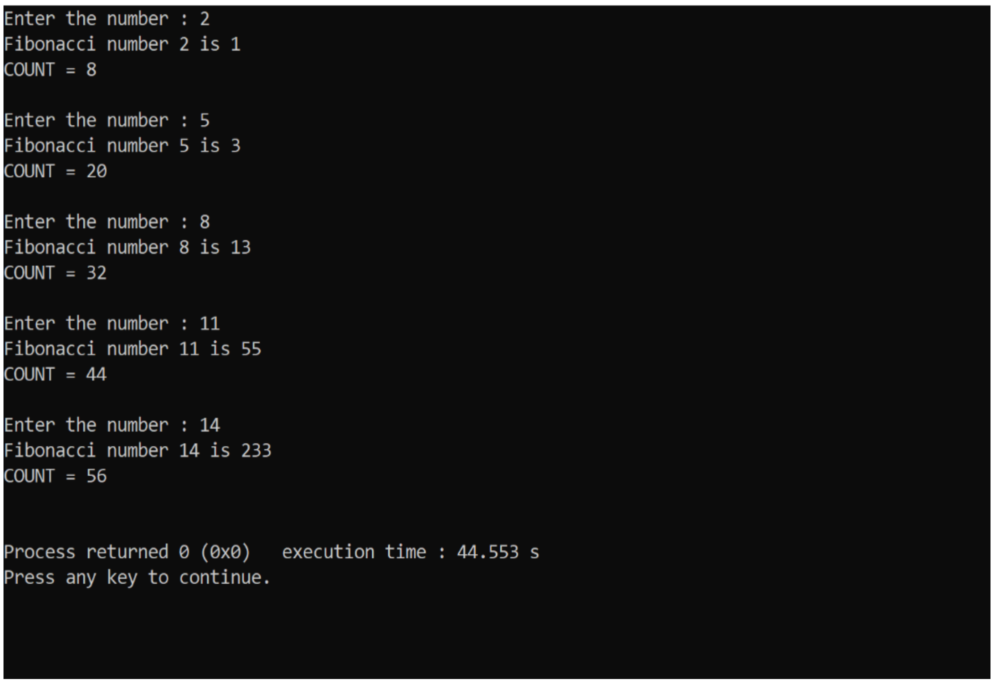
Ques (b): Count number of steps for **FIBONACCI – ALGORITHM**.

Code:

```
#include<iostream>

using namespace std;

int count = 0 ;

void fibonacci (int n)

{

   ::count++ ; // if-else check

   if (n<=1)

   {

         cout << n ;

         ::count++ ;

   }


   else

   {

      int fnm1 = 0 , fnm2 = 1 , fn ;   ::count++ ;


      for (int i=2 ; i<=n ; i++)

      {

         ::count++ ; // for loop


         fn = fnm1 + fnm2 ;    ::count++ ;


         fnm2 = fnm1 ;   ::count++ ;
```

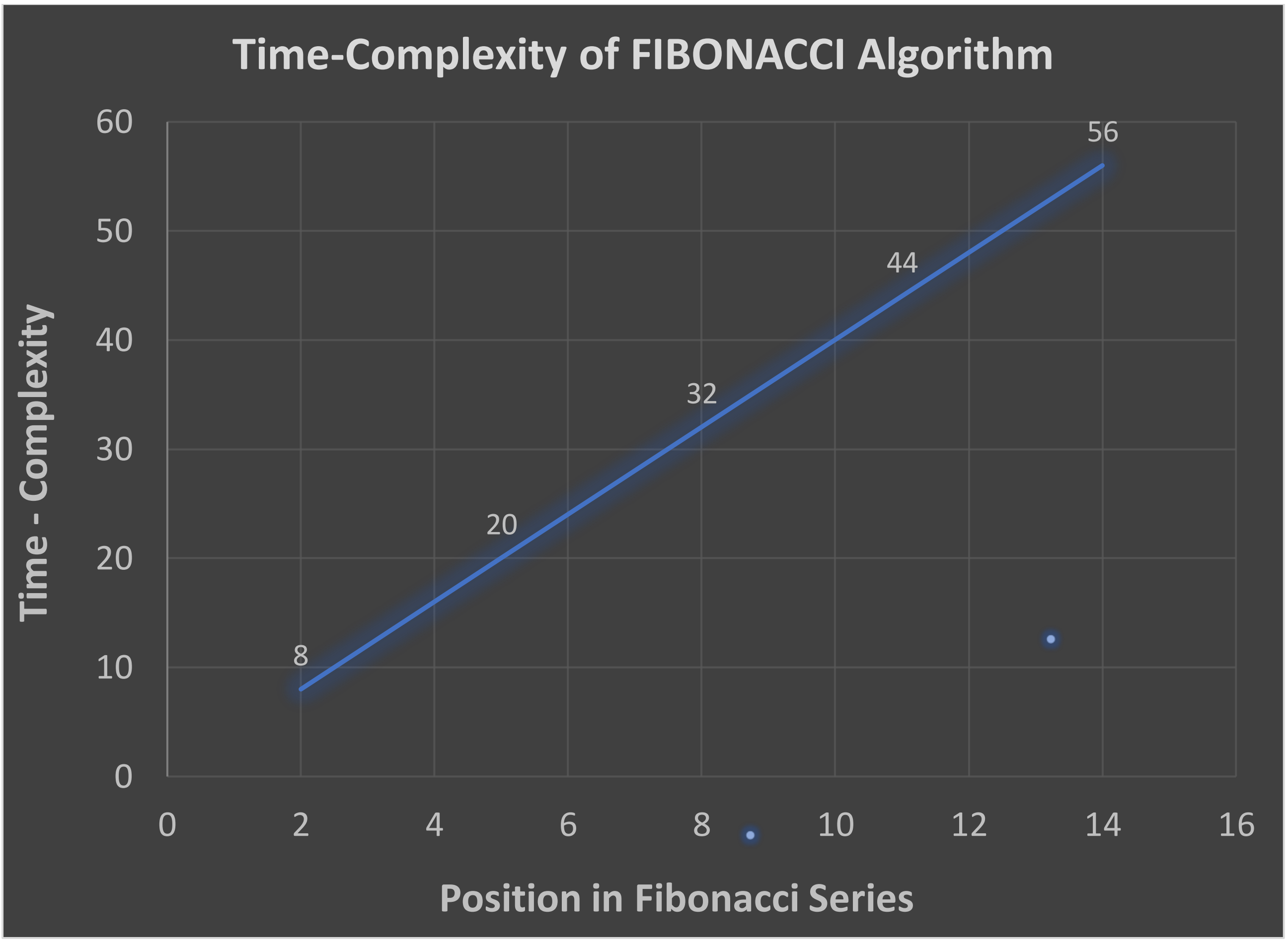
```
          fnm1 = fn ;    ::count++ ;

       }

       ::count++ ; // since condition of for loop is checked n+1 times

       cout << "Fibonacci number " << n <<" is "  << fn << "\n" ;
::count++ ;
   }
}

int main()
{
  for (int i=0 ; i<5 ; i++)
  {
       int n ;
       cout << "Enter the number : " ;
       cin >> n ;

       fibonacci(n) ;
       cout << "COUNT = " << ::count << "\n\n" ;
       ::count = 0 ;
  }
}
```

Output:


```
Enter the number : 2
Fibonacci number 2 is 1
COUNT = 8

Enter the number : 5
Fibonacci number 5 is 3
COUNT = 20

Enter the number : 8
Fibonacci number 8 is 13
COUNT = 32

Enter the number : 11
Fibonacci number 11 is 55
COUNT = 44

Enter the number : 14
Fibonacci number 14 is 233
COUNT = 56


Process returned 0 (0x0)   execution time : 44.553 s
Press any key to continue.
```

Graph:

Position in Fibonacci Series	Time-Complexity
2	8
5	20
8	32
11	44
14	56


Time-Complexity of FIBONACCI Algorithm
Time - Complexity
Position in Fibonacci Series
0
10
20
30
40
50
60
0
2
4
6
8
10
12
14
16
8
20
32
44
56

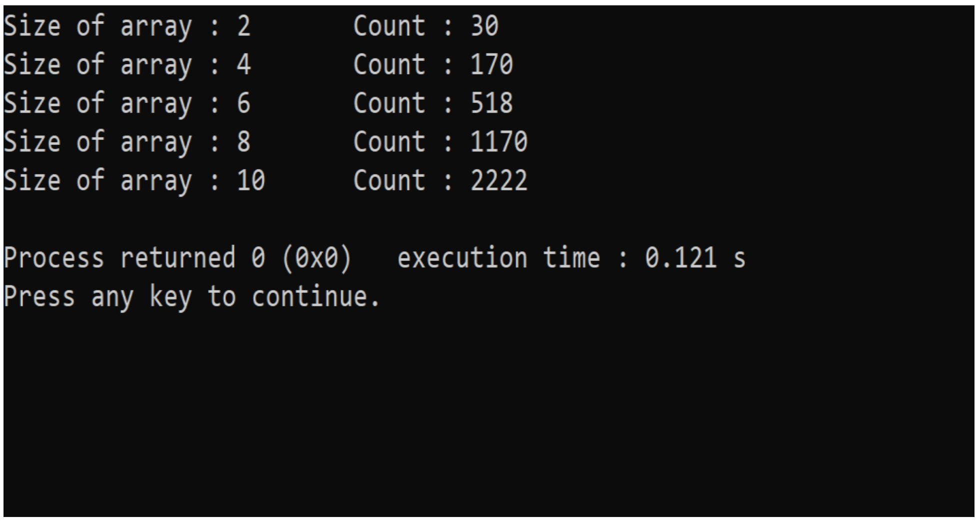
Ques (c): Count number of steps for **MATRIX-MULTIPLICATION – ALGORITHM**.

Code:

```
#include<iostream>
#define N 10
using namespace std;

int matrixMultiplication (int A[N][N] , int B[N][N] , int C[N][N] , int s)
{
   int count = 0 ;

   for (int i=0 ; i<s ; i++)
   {
      count++ ;
      for (int j=0 ; j<s ; j++)
      {
         count++ ;
         for (int k=0 ; k<s ; k++)
         {
            count++ ;

            C[i][j] += A[i][k] * B[k][j] ;   count++ ;
         }
         count++ ; // since k loop's condition will be checked s+1 times
```

```
        }
        count++ ; // since j loop's condition will be checked s+1 times
    }
    count++ ; // since i loop's condition will be checked s+1 times

    count++ ;
    cout << "Count : " <<  count << "\n" ;
}

int main()
{
    int arr1[N][N] = {0} , arr2[N][N] = {0} , res[N][N] = {0} ;
    for (int i=2 ; i<=N ; i=i+2)
    {
        cout << "Size of array : " << i << "\t" ;
        matrixMultiplication(arr1 , arr2 , res , i) ;
    }
}
```

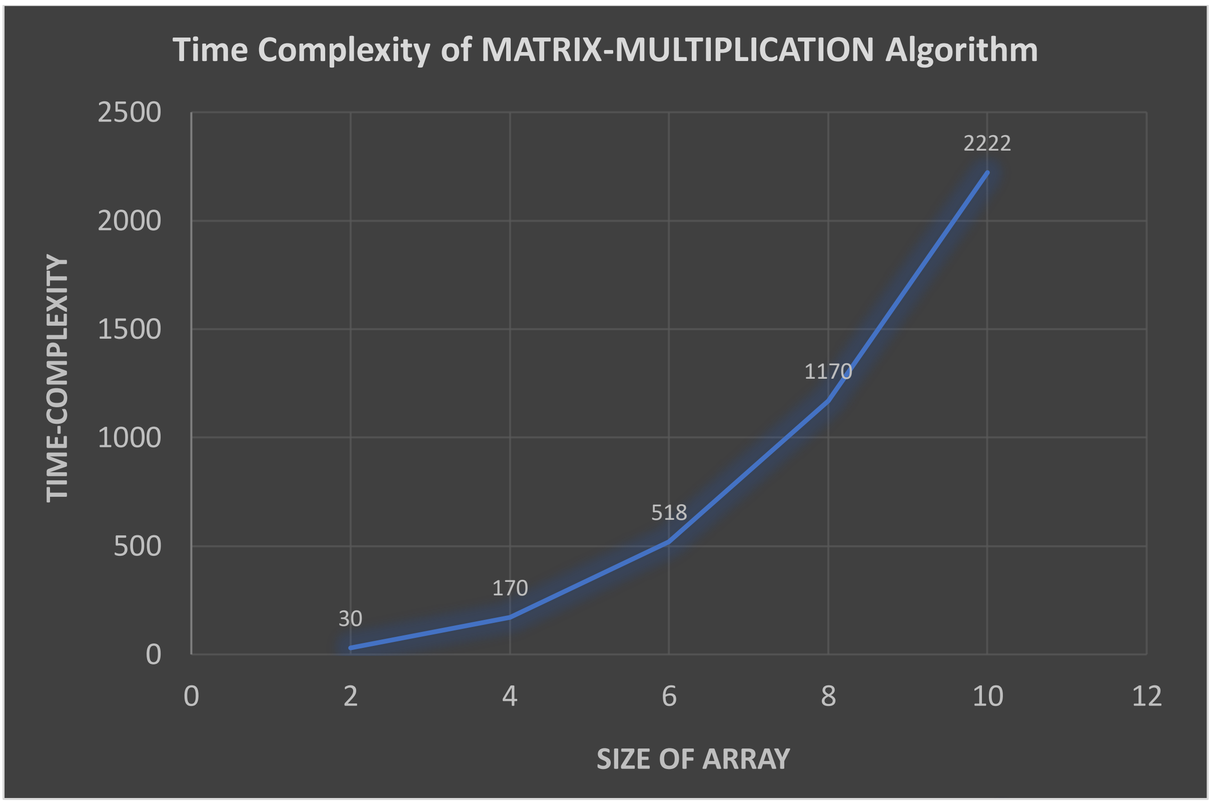
Output:



```
Size of array : 2       Count : 30
Size of array : 4       Count : 170
Size of array : 6       Count : 518
Size of array : 8       Count : 1170
Size of array : 10      Count : 2222

Process returned 0 (0x0)   execution time : 0.121 s
Press any key to continue.
```

Graph:

Size of Array	Time Complexity
2	30
4	170
6	518
8	1170
10	2222


Time Complexity of MATRIX-MULTIPLICATION Algorithm
TIME-COMPLEXITY
SIZE OF ARRAY
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12
30
170
518
1170
2222