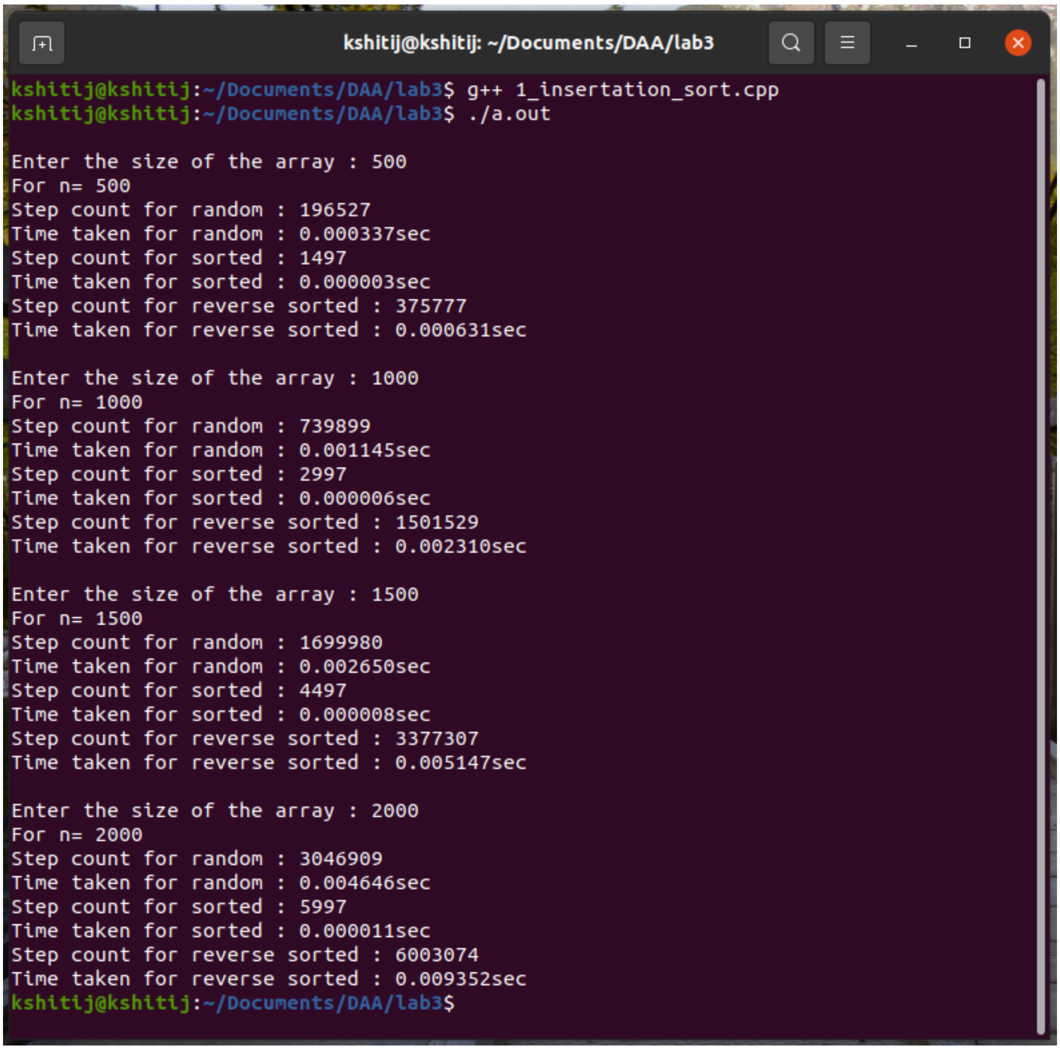
DAA Lab-3

Name: Kshitij Kumar Sharma   Roll No.: 1905514   Date: 29/07/2021

Q1) Write a program to sort a given set of elements using the insertion sort. Additionally, determine the time required (in terms of steps) to sort the elements.
(Note: assume cost of any basic operation is 1, i.e., c1= c2 = ... = c8 = 1).
1) Repeat the experiment for different values of n = 500, 1000, 5000, 10000
2) For each of aforementioned case, consider arrays as sorted, random, and reverse-sorted. Provide the complexity in terms of step count
Program:

```
/*
Written by: Kshitij Kumar Sharma                    Roll No: 1905514
Question:
Write a program to sort a given set of elements using the insertion sort. Additionally, determine the time required (in terms of steps) to sort the elements.
(Note: assume cost of any basic operation is 1, i.e., c1= c2 = ... = c8 = 1).
1) Repeat the experiment for different values of n = 500, 1000, 5000, 10000
2) For each of aforementioned case, consider arrays as sorted, random, and reverse-sorted. Provide the complexity in terms of step count

Ideal of the Solution:
First I will define a insertation sort function and then I will generate random array of different size and then I will check for         all the three cases.
*/
#include<bits/stdc++.h>
#include<stdio.h>
#include<time.h>
#include<stdlib.h>

using namespace std;

int insertationSort(int a[],int n)        //Insertion sort function taking an array and its size as parameter
        {
        int i,j,temp,c=0;                    //c is the step counter
        for(i=1;i<n;i++)
         {
        temp=a[i];                           //storing the element into a temporary variable for finding it position
         c++;
         j=i-1;
         c++;
        while(temp<a[j])                             //loop for finding the position
          {
          c++;
        if(j==-1)
        {
          c++;
          break;
        }
        a[j+1]=a[j];                                 //swaping for creating position
        c++;
        j--;
        c++;
    }
    a[j+1]=temp;                                //Inserting the element at the position
    c++;
  }
  return c;                                     //returning the total step count
}

int main()
        {
         int n,i,j,k,c1,c2,c3;                  //Variables for keeping different step counts
         clock_t start, end;                    //Variables for keeping start and end time
        double cpu_time_used;                   //Variable for keeping cpu time used
         for(i=0;i<4;i++)
                {
                cout<<endl;
                cout<<"Enter the size of the array : ";
                cin>>n;
                int a[n];
                srand(time(0));
                for(j=0;j<n;j++)
                        a[j]=rand()%1000000;            //generating the random array
                cout<<"For n= "<<n<<endl;
                start=clock();                          //keeping start time of the clock for random array
                c1=insertationSort(a,n);                //sorting
                end=clock();                            //keeping end time of the clock for random array
                cpu_time_used=((double)(end-start))/CLOCKS_PER_SEC; //calculating cpu time used for random array
                cout<<"Step count for random : "<<c1<<endl;
                //cout<<"time taken : "<<cpu_time_used<<endl;
                printf("Time taken for random : %fsec \n",cpu_time_used);
                sort(a,a+n);

                start=clock();                          //keeping start time of the clock for sorted array
                c2=insertationSort(a,n);                //sorting
                end=clock();                            //keeping end time of the clock for sorted array
                cpu_time_used=((double)(end-start))/CLOCKS_PER_SEC;//calculating cpu time used for sorted array
                cout<<"Step count for sorted : "<<c2<<endl;
                //cout<<"time taken : "<<cpu_time_used<<endl;
                printf("Time taken for sorted : %fsec \n",cpu_time_used);

                sort(a,a+n,greater<int>());
                start=clock();                          //keeping start time of the clock for reverse sorted array
                c3=insertationSort(a,n);                //sorting
                end=clock();                            //keeping end time of the clock for reverse sorted array
                cpu_time_used=((double)(end-start))/CLOCKS_PER_SEC;  //calculating  cpu  time  used  for  reverse
sorted array
                cout<<"Step count for reverse sorted : "<<c3<<endl;
                //cout<<"time taken : "<<cpu_time_used<<endl;
                printf("Time taken for reverse sorted : %fsec \n",cpu_time_used);
                }
        }
```

Output:



```
kshitij@kshitij:~/Documents/DAA/lab3$ g++ 1_insertation_sort.cpp
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out

Enter the size of the array : 500
For n= 500
Step count for random : 196527
Time taken for random : 0.000337sec
Step count for sorted : 1497
Time taken for sorted : 0.000003sec
Step count for reverse sorted : 375777
Time taken for reverse sorted : 0.000631sec

Enter the size of the array : 1000
For n= 1000
Step count for random : 739899
Time taken for random : 0.001145sec
Step count for sorted : 2997
Time taken for sorted : 0.000006sec
Step count for reverse sorted : 1501529
Time taken for reverse sorted : 0.002310sec

Enter the size of the array : 1500
For n= 1500
Step count for random : 1699980
Time taken for random : 0.002650sec
Step count for sorted : 4497
Time taken for sorted : 0.000008sec
Step count for reverse sorted : 3377307
Time taken for reverse sorted : 0.005147sec

Enter the size of the array : 2000
For n= 2000
Step count for random : 3046909
Time taken for random : 0.004646sec
Step count for sorted : 5997
Time taken for sorted : 0.000011sec
Step count for reverse sorted : 6003074
Time taken for reverse sorted : 0.009352sec
kshitij@kshitij:~/Documents/DAA/lab3$
```

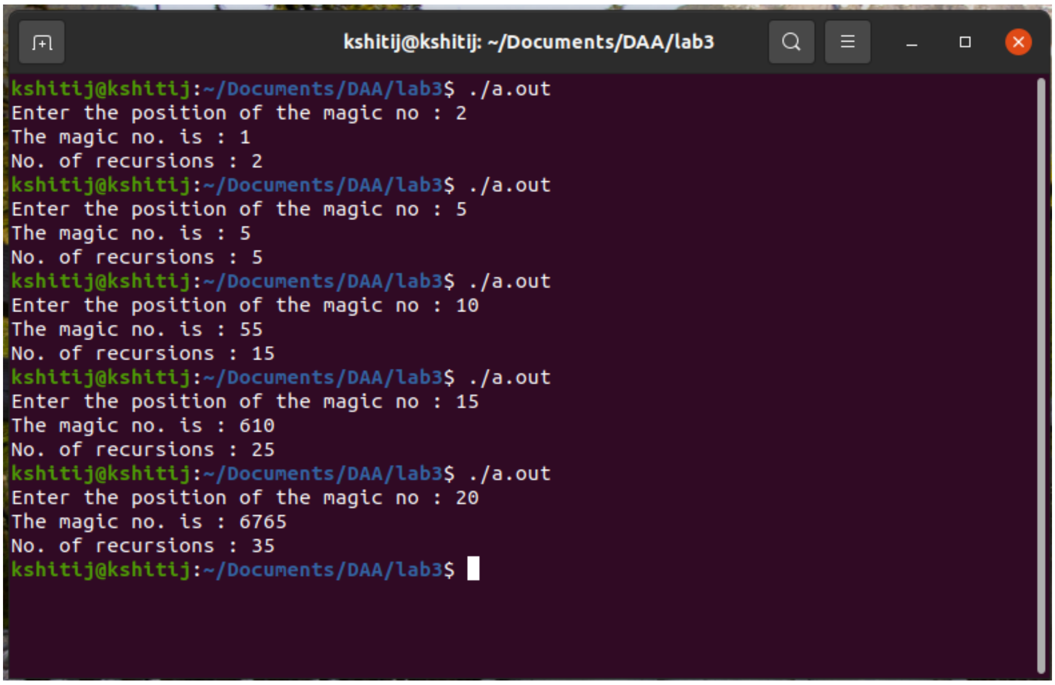
Q2) Write a program to compute the nth Magic number (recursively) defined as below and find its time complexity (in terms of number of recursions).nth magic number MN(n) = MN(n-1) + MN(n-2), whereas MN(0) = 0, and MN(1) = 1.

Program:

```
/*
Written By: Kshitij Kumar Sharma          Roll No: 1905514
Question:
Write a program to compute the nth Magic number (recursively) defined as below and find its time complexity (in terms of number of recursions).nth magic number MN(n) = MN(n-1) + MN(n-2), whereas MN(0) = 0, and MN(1) = 1.

Idea of the solution:
I am defining a fibo function for calculating the n th fibonachi series number recursively, and then applying the divide and conquer approach for finding the n th number.
*/
#include<bits/stdc++.h>

using namespace std;

int t=0;
                                                //Variable for keeping no. of recursions

int fibo(int n,int a,int b,int c)               //function for calculating the nth fibonachi number.
```

```
        {
        t++;
        if(n==0)
                return a;
        if(n==1)
                return b;
        if(n==2)                                //break condition of the recursion
                return c;
        a=b;
        b=c;
        c=a+b;
        n--;
        fibo(n,a,b,c);                          //recursively calling the fibo function with the updated value
        }

int main()
        {
         int n,mn;
         cout<<"Enter the position of the magic no : ";
         cin>>n;
        //taking the n th number from user
         if(n>=2)
                mn=fibo(n-1,0,1,1)+fibo(n-2,0,1,1);                 //applying divide and conquer
         cout<<"The magic no. is : "<<mn<<endl;
         cout<<"No. of recursions : "<<t<<endl;
        }
```

Output:



```
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out
Enter the position of the magic no : 2
The magic no. is : 1
No. of recursions : 2
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out
Enter the position of the magic no : 5
The magic no. is : 5
No. of recursions : 5
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out
Enter the position of the magic no : 10
The magic no. is : 55
No. of recursions : 15
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out
Enter the position of the magic no : 15
The magic no. is : 610
No. of recursions : 25
kshitij@kshitij:~/Documents/DAA/lab3$ ./a.out
Enter the position of the magic no : 20
The magic no. is : 6765
No. of recursions : 35
kshitij@kshitij:~/Documents/DAA/lab3$ 
```