ALGORITHMS LABORATORY
[CS-2098]
Individual Work

Lab. No.- 02 Date.20-7-21

Roll Number:	1929039	Branch/Section:	CSCE
Name in Capital:	SAMBARAN SENGUPTA		

Program No: 2.1

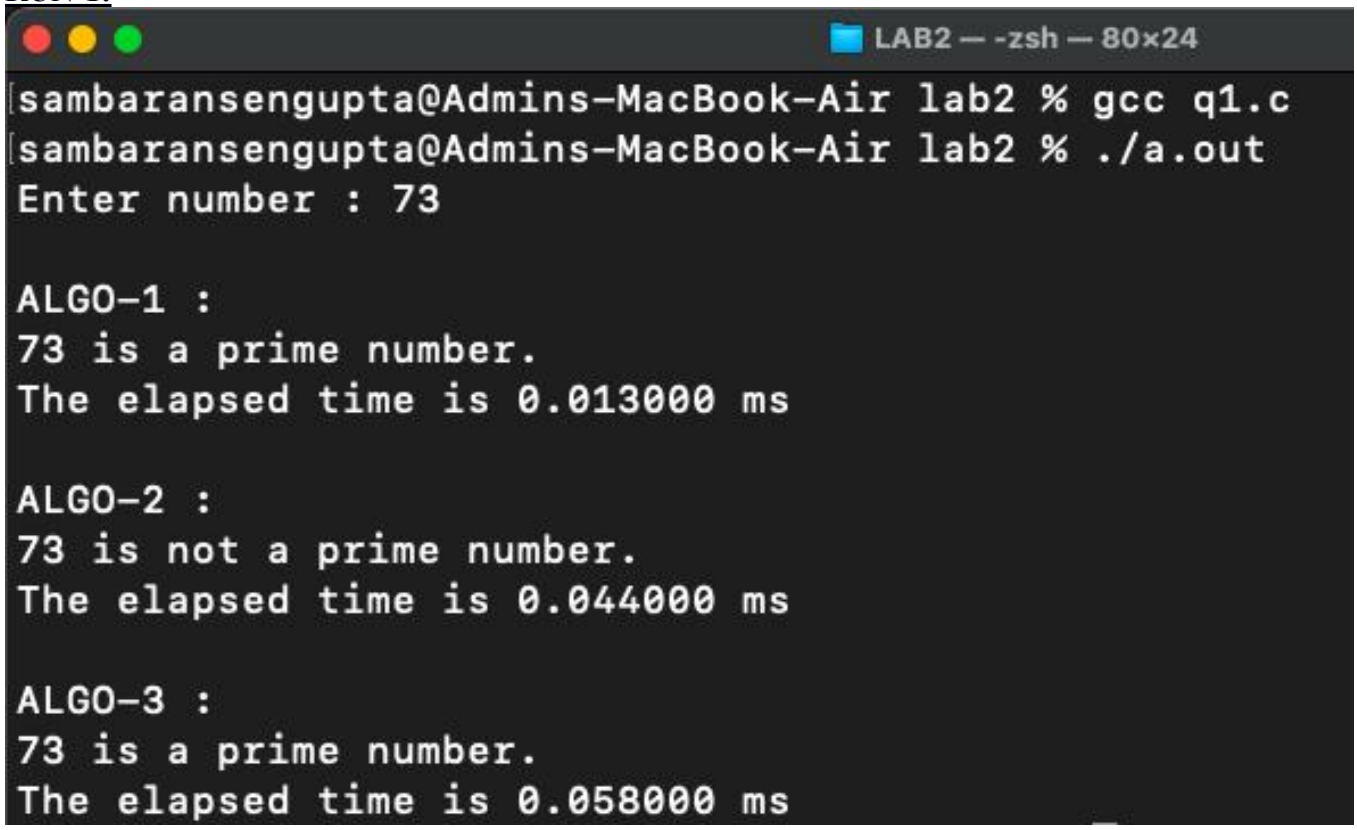
Program Title:

Write a program to test whether A number n, entered through keyboArd is prime or not by using different algorithms you know for atleast 10 inputs and note down the time complexity by step/frequency count method for each algorithm & for each input.

Finally make a comparison of time complexities found for different inputs, plot an appropriate graph & decide which algorithm is faster.

Input/Output Screenshots:

RUN-1:



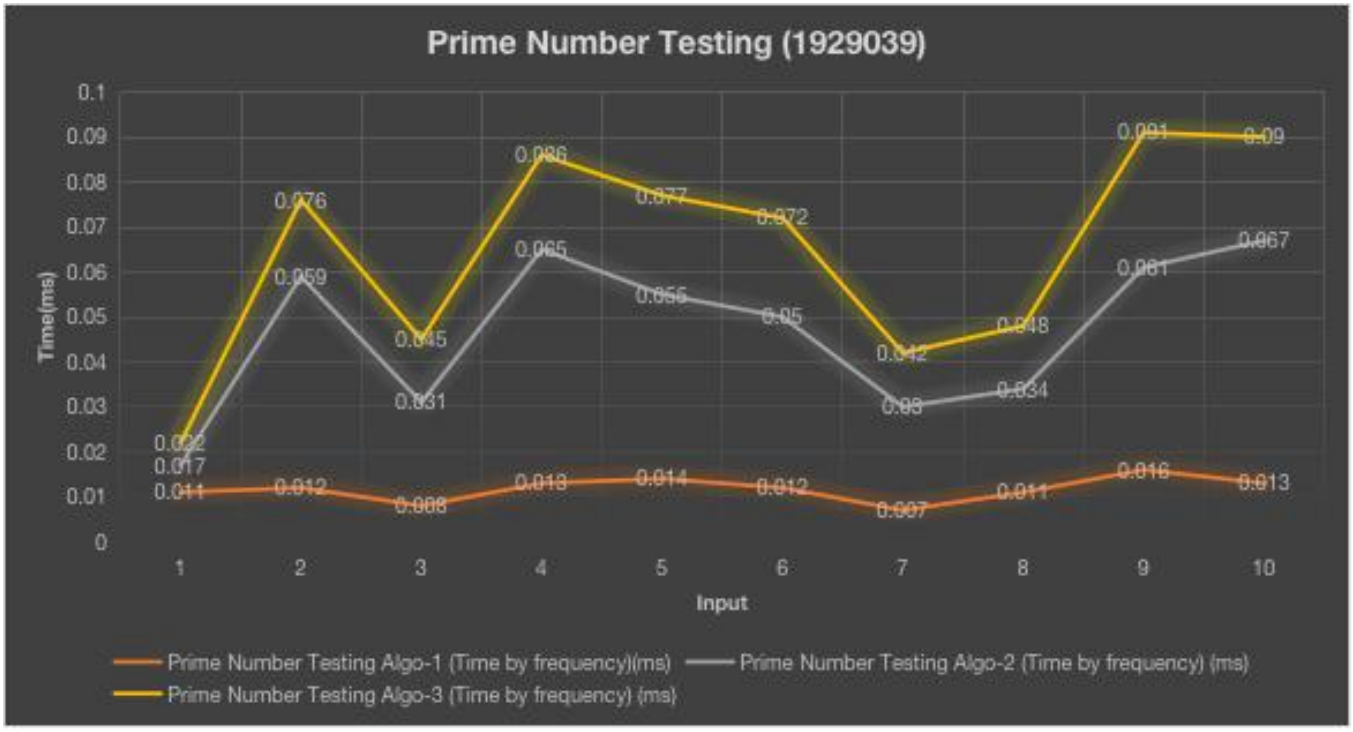
RUN-2

```
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Enter number : 100

ALGO-1 :
100 is not a prime number.
The elapsed time is 0.014000 ms

ALGO-2 :
100 is not a prime number.
The elapsed time is 0.056000 ms

ALGO-3 :
100 is not a prime number.
The elapsed time is 0.080000 ms
```

TABLE:

Sl No.	Input(n)	Prime Number Testing		
		Algo-1 (Time by frequency) (ms)	Algo-2 (Time by frequency) (ms)	Algo-3 (Time by frequency) (ms)
1	n=2	0.011	0.017	0.022
2	n=5	0.012	0.059	0.076
3	n=10	0.008	0.031	0.045
4	n=15	0.013	0.065	0.086
5	n=19	0.014	0.055	0.077
6	n=25	0.012	0.05	0.072
7	n=51	0.007	0.03	0.042
8	n=503	0.011	0.034	0.048
9	n=751	0.016	0.061	0.091
10	n=1001	0.013	0.067	0.09

GRAPH:



Source code

```
#include <stdio.h>
#include <stdbool.h>//to use bool, true, false
#include <time.h>
#include <math.h>
#include <stdlib.h>
bool checkPrimality1(int n){
    if(n<=1){
        return false;
    }
    for(int i=2; i<=sqrt(n); i++){
        if(n%i==0){
            return false;
        }
    }
    return true;
}
bool checkPrimality2(int n){
    if(n==1) return false;
            int a = 2;
            while(a<n-1){
        int x = pow(a,n-1);
                        if(x%n != 1){
                                return false;
                        }
                        a++;
            }
            return true;
}
```

```
int powerhelper(int x, unsigned int y, int p){
	int res = 1;
	x = x % p;

	while (y > 0){
		if (y & 1)
			res = (res*x) % p;

		y = y/2;
		x = (x*x) % p;
	}
	return res;
}

bool miillerTest(int d, int n){

	int a = 2 + rand() % (n - 4);

	int x = powerhelper(a, d, n);

	if (x == 1 || x == n-1)
	return true;

	while(d != n-1){
		x = (x * x) % n;
		d *= 2;

		if (x == 1)	return false;
		if (x == n-1) return true;
	}


	return false;
}

bool checkPrimality3(int n, int k){

	if(n <= 1 || n == 4) return false;
	if(n <= 3) return true;

	int d = n - 1;
	while(d % 2 == 0)
		d = d/2;

	for(int i = 0; i < k; i++)
		if(!miillerTest(d, n))
			return false;
	return true;
}
```

```
int main() {
   double time_spent = 0.0;
   int n;
   printf("Enter number : ");
   scanf("%d", &n);
   printf("\nALGO-1 : \n");
   clock_t begin = clock();
   if(checkPrimality1(n))
      printf("%d is a prime number.\n", n);
   else
      printf("%d is not a prime number.\n", n);
   clock_t end = clock();
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("The elapsed time is %f ms\n", time_spent*1000);

   printf("\nALGO-2 : \n");
   begin = clock();
   if(checkPrimality2(n))
      printf("%d is a prime number.\n", n);
   else
      printf("%d is not a prime number.\n", n);

   end = clock();
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("The elapsed time is %f ms\n", time_spent*1000);

   printf("\nALGO-3 : \n");
   begin = clock();
   if(checkPrimality3(n, 4))//4 iterations
      printf("%d is a prime number.\n", n);
   else
      printf("%d is not a prime number.\n", n);

   end = clock();
   time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
   printf("The elapsed time is %f ms\n", time_spent*1000);


}
```

Conclusion/Observation

We observe that Algo-1 is faster than the other two.

Program No: 2.2

Program Title:

Write a program to find out GCD (greAtest common divisor) using the following three

algorithms.

a) Euclid's algorithm

b) Consecutive integer checking algorithm.

c) Middle school procedure which makes use of common prime factors.

For finding list of primes implement sieve of Eratosthenes algorithm.

Write a program to find out which algorithm is faster for the following data.

Estimate how many times it will be faster than the other two by

step/frequency count method in each case.

i.Find GCD of two numbers when both Are very lArge i.e.GCD(31415, 14142) by applying each of the above algorithms.

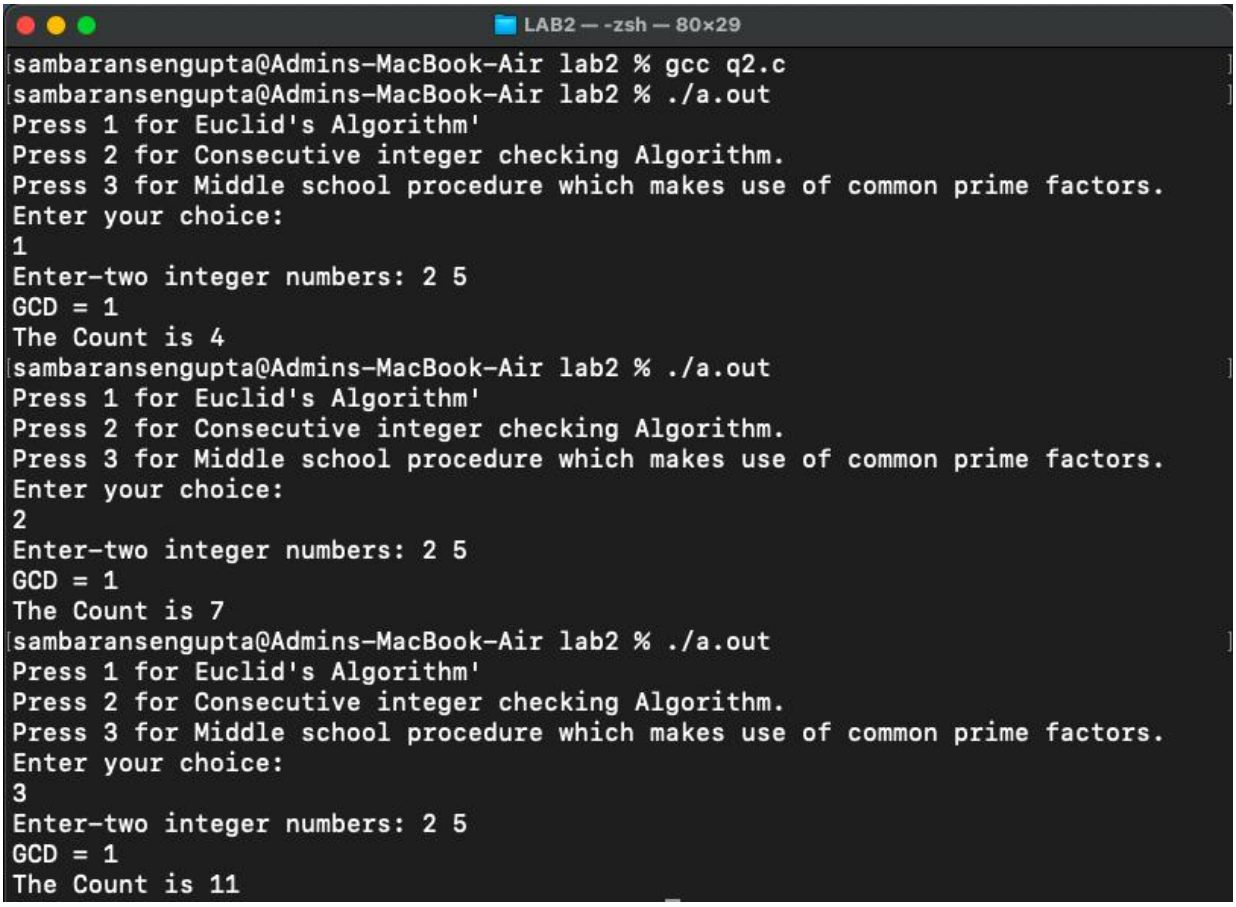
ii.Find GCD of two numbers when one of them cAn be very lArge i.e.GCD(56, 32566) or GCD(34218, 56) by applying each of the above algorithms.

iii. Find GCD of two numbers when both Are very smAll i.e.GCD(12,15) by applying each of the above algorithms.

iv. Find GCD of two numbers when both Are sAme i.e.GCD(31415, 31415) or GCD(12, 12) by applying each of the above algorithms.

Input/Output Screenshots:

RUN-1:


LAB2 — -zsh — 80×29
sambaransengupta@Admins-MacBook-Air lab2 % gcc q2.c
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Press 1 for Euclid's Algorithm'
Press 2 for Consecutive integer checking Algorithm.
Press 3 for Middle school procedure which makes use of common prime factors.
Enter your choice:
1
Enter-two integer numbers: 2 5
GCD = 1
The Count is 4
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Press 1 for Euclid's Algorithm'
Press 2 for Consecutive integer checking Algorithm.
Press 3 for Middle school procedure which makes use of common prime factors.
Enter your choice:
2
Enter-two integer numbers: 2 5
GCD = 1
The Count is 7
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Press 1 for Euclid's Algorithm'
Press 2 for Consecutive integer checking Algorithm.
Press 3 for Middle school procedure which makes use of common prime factors.
Enter your choice:
3
Enter-two integer numbers: 2 5
GCD = 1
The Count is 11

RUN-2:

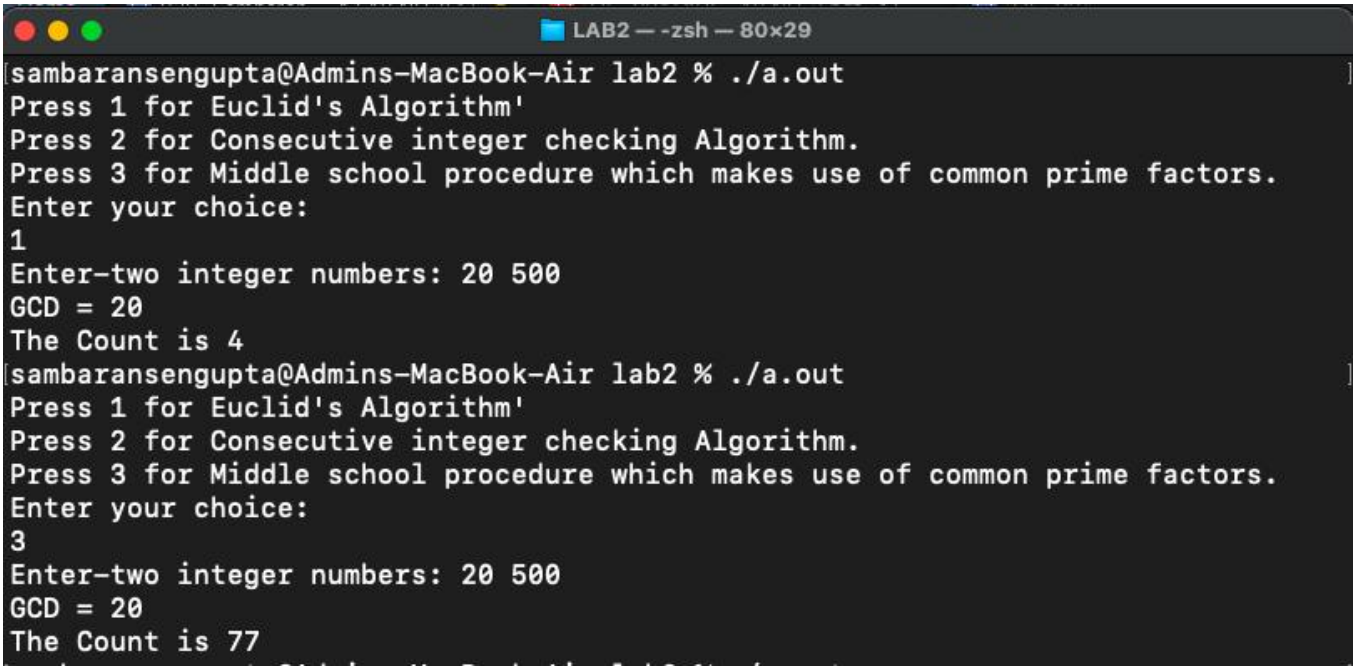

LAB2 — -zsh — 80×29
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Press 1 for Euclid's Algorithm'
Press 2 for Consecutive integer checking Algorithm.
Press 3 for Middle school procedure which makes use of common prime factors.
Enter your choice:
1
Enter-two integer numbers: 20 500
GCD = 20
The Count is 4
sambaransengupta@Admins-MacBook-Air lab2 % ./a.out
Press 1 for Euclid's Algorithm'
Press 2 for Consecutive integer checking Algorithm.
Press 3 for Middle school procedure which makes use of common prime factors.
Enter your choice:
3
Enter-two integer numbers: 20 500
GCD = 20
The Count is 77

TABLE:

Sl No.	Input GCD(x,y)	GCD Algorithm			Remarks
		Euclid's Algorithm	Consecutive Integer Checking Algorithm	Middle School procedure Algorithm	
1	GCD(3145,1412)	4	42427	424446	Euclid's algorithm is faster
2	GCD(56,32566)	4	166	185	Euclid's algorithm is faster
3	GCD(34218,56)	4	166	188	Euclid's algorithm is faster
4	GCD(12, 15)	4	31	50	Euclid's algorithm is faster
5	GCD(31415, 31415)	4	6	94256	Euclid's algorithm is faster
6	GCD(12, 12)	4	6	47	Euclid's algorithm is faster

Source code

```
#include<stdio.
h>
#include<stdlib.
h>

int main() {

        int choice, num1,num2,r,min,i,c,ans=1,m,n,count;

        printf("Press 1 for Euclid's Algorithm'\n");

        printf("Press 2 for Consecutive integer checking Algorithm.\n");

        printf("Press 3 for Middle school procedure which makes use of common prime factors.\n");
        printf("Enter your choice:\n");

        scanf("%d",&choic
        e); switch (choice)
        {

        case 1: {

                printf("Enter-two integer numbers:
                "); count++;

   scanf ("%d %d", &num1,
   &num2); count++;

   while (num2 > 0) {

      r = num1 %
      num2; num1=
      num2; num2 = r;

   }

   count++;

    printf("GCD = %d
    \n",num1); count++;
```

```
 printf("The Count is %d\n",count);

                break;

}

        case 2:{

                printf("Enter-two integer numbers:
                "); count++;

    scanf ("%d %d", &num1,
    &num2); count++;

            if(num1<nu
      m2) min=num1;

    else

       min=nu
       m2;
       count++;
       r=min;
       count++;

    while (r>0){

          if(num1 % r == 0 && num2 % r ==
       0) return r;

       count
    ++; r=r -
    1;
    count++;

        printf ("GCD = %d \n",r);
        count++;

        printf("The Count is %d\n",count);

          }

          break;

          }

          case 3:{

                printf("Enter-two integer numbers:
                "); count++;

    scanf ("%d %d", &num1,
          &num2); count++;
```

```
    c=num1>=num2?num2:nu
    m1; count++;

    for(i=2;i<=c;i+
          +){ m=0;

          n=0;

          if(num1%i==0){

                m=1;

                num1=num1/i;

                }

                count++;
                if(num2%i==
                0){n=1;

                      num2=num2/i;

                }count++;

                if(n==1 && m==1){

                      ans=ans*i;

                }

                count++;
                if(n==1 ||
                m==1) i--;

          }count++;

           printf ("GCD = %d \n",ans);

           count++;

           printf("The Count is
                %d\n",count);

                break;

          }

          default: printf("wrong Input\n");

}

}
```

Conclusion/Observation

We are able to observe and compare the difference in speed and execution of the algorithms.

Program No: 2.3

Program Title:

Write a menu driven program as given below, to sort an array of n integers in ascending order by insertion sort Algorithm and determine the time required (in terms of step/frequency count) to sort the elements. Repeat the experiment for different values of n and different nature of data (i.e.apply insertion sort algorithm on the data of array that are already sorted, reversely sorted and random data). Finally plot a graph of the time taken versus n for each type of data. The elements can be read from a file or can be generated using the random number generator.

Input/Output
Screenshots: RUN-1:

```
LAB2 — -zsh — 97×39
Enter your choice:
2
8       50      74      59      31      73      45      79      24      10
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
3
Array sorted in ascending order
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
2
8       10      24      31      45      50      59      73      74      79
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
4
Array sorted in descending order
```

```
LAB2 — -zsh — 97×39
4
Array sorted in descending order
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
2
79      74      73      59      50      45      31      24      10      8
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
5
Time complexity to sort random data: 0.000005
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
6
Time complexity to sort random data: 0.000004
```

```
Enter your choice:
6
Time complexity to sort random data: 0.000004
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
7
Time complexity to sort random data: 0.000007
**MAIN MENU**
0. Quit
1. Input n random numbers into array
2. Display array
3. Sort array in ascending order
4. Sort array in descending order
5. Time complexity to sort random data in ascending order
6. Time complexity to sort data in ascending order which is already sorted in ascending order
7. Time complexity to sort data in ascending order which is already sorted in descending order
Enter your choice:
0
```

Source code

```
#include<stdio.h>
#include<time.h>
#include<stdlib.h>
void input(int arr[], int n){
   for(int i=0; i<n; i++){
      arr[i]=1+rand()%100;
   }
}
void InsertionSortAsc(int arr[], int n){ int i, j, key;
   for (i=1; i<n; i++){
      key=arr[i];
      j=i-1;
      while (j>=0 && arr[j]>key){ arr[j+1]=arr[j];
         j=j-1;
      }
      arr[j+1]=key;
   }
}

void InsertionSortDesc(int arr[], int n){ int i, j, key;
   for (i=1; i<n; i++){
      key=arr[i];
      j=i-1;
      while (j>=0 && arr[j]<key){
         arr[j+1]=arr[j];
         j=j-1;
      }
      arr[j+1]=key;
   }
```

```
}
void display(int arr[], int n){
    for (int i=0; i<n; i++)
        printf("%d\t", arr[i]);
    printf("\n");
}


int main(){
    int n, option=0, arr[50000]; time_t strt, end;
    double t;
    do {
        printf("**MAIN MENU**\n"); printf("0. Quit\n");
        printf("1. Input n random numbers into array\n"); printf("2. Display array\n");
        printf("3. Sort array in ascending order\n"); printf("4. Sort array in descending order\n");
        printf("5. Time complexity to sort random data in ascending order\n");
        printf("6. Time complexity to sort data in ascending order which is already sorted in ascending order\n");
        printf("7. Time complexity to sort data in ascending order which is already sorted in descending
order\n");
        printf("Enter your choice: \n"); scanf("%d", &option);
        switch (option){
            case 1:
                printf("Enter n\n"); scanf("%d", &n);
                input(arr, n);
                break;
            case 2:
                display(arr, n);
                break;
            case 3:
                InsertionSortAsc(arr, n);
                printf("Array sorted in ascending order \n");
```

```
            break;
        case 4:
            InsertionSortDesc(arr, n);
            printf("Array sorted in descending order \n"); break;
        case 5:
            strt=clock(); InsertionSortAsc(arr, n); end=clock();
            t=end-strt;
            printf("Time complexity to sort random data: %f\n", t/CLOCKS_PER_SEC); break;
        case 6:
            InsertionSortAsc(arr, n); strt=clock(); InsertionSortAsc(arr, n); end=clock();
            t=end-strt;
            printf("Time complexity to sort random data: %f\n", t/CLOCKS_PER_SEC); break;
        case 7:
            InsertionSortDesc(arr, n); strt=clock(); InsertionSortAsc(arr, n); end=clock();
            t=end-strt;
            printf("Time complexity to sort random data: %f\n", t/CLOCKS_PER_SEC); break;
        }
    } while(option!=0);
}
```

Conclusion/Observation

We're able to observe time complexity of insertion sort.