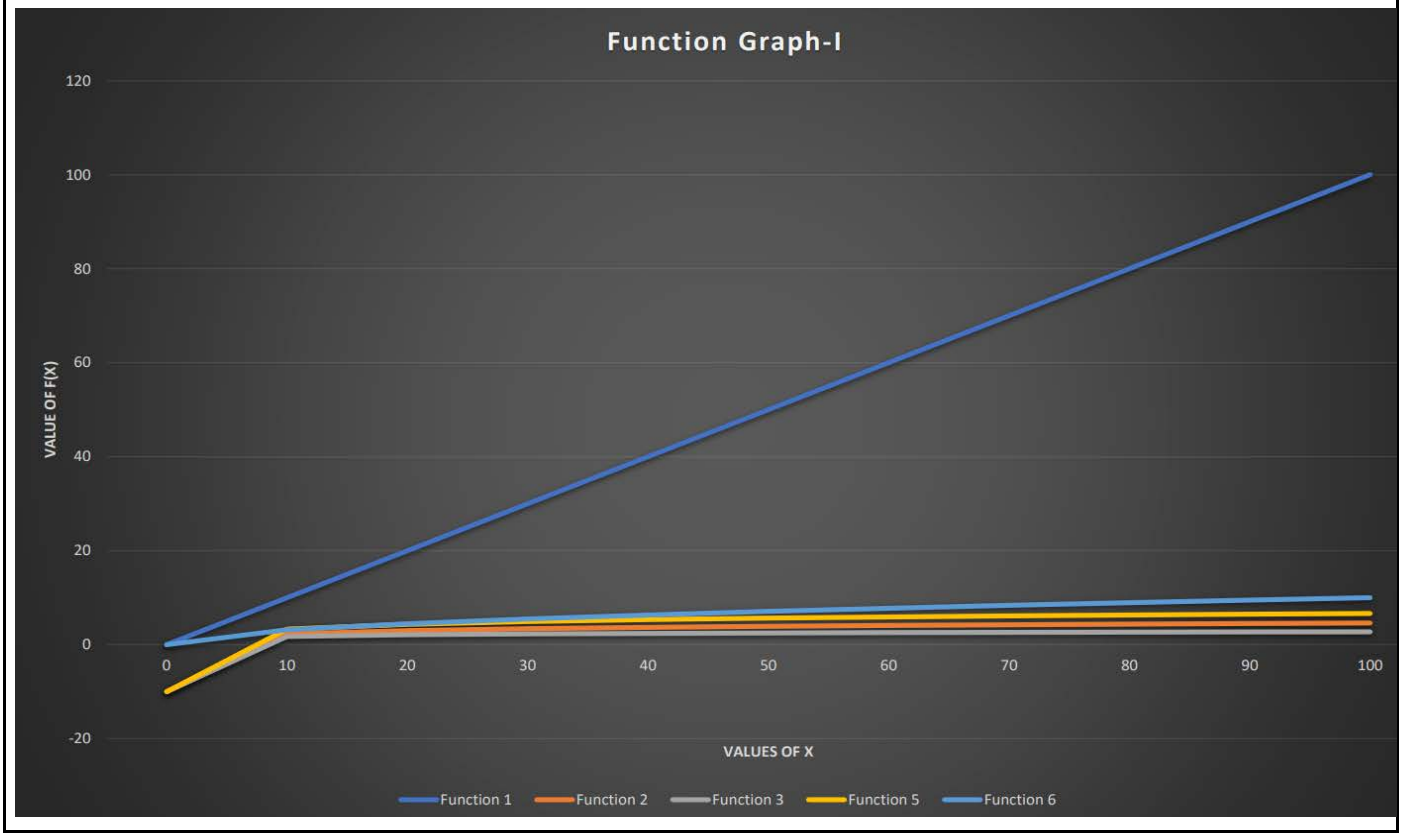
Name	Ronak Thakar
UID no.	2021700066
Experiment No.	1

AIM:	1-A: To implement the various functions e.g. linear, non-linear, quadratic, exponential etc. 1-B: Experiment on finding the running time of an algorithm.
PART 1-A Program 1	
PROBLEM STATEMENT :	For this experiment, we have to implement at least 10 functions from the list. The input (i.e. n) to all the above functions varies from 0 to 100 with increment of 10. Then add the function n! in the list and execute the same for n from 0 to 20 with increment of 2.
ALGORITHM/ THEORY:	In this experiment part 1-A we must use 10 functions from the given list write a simple C code for it and then use the output from the code to draw a graph for the 10 functions + the factorial function and draw our conclusion regarding the
PROGRAM:	

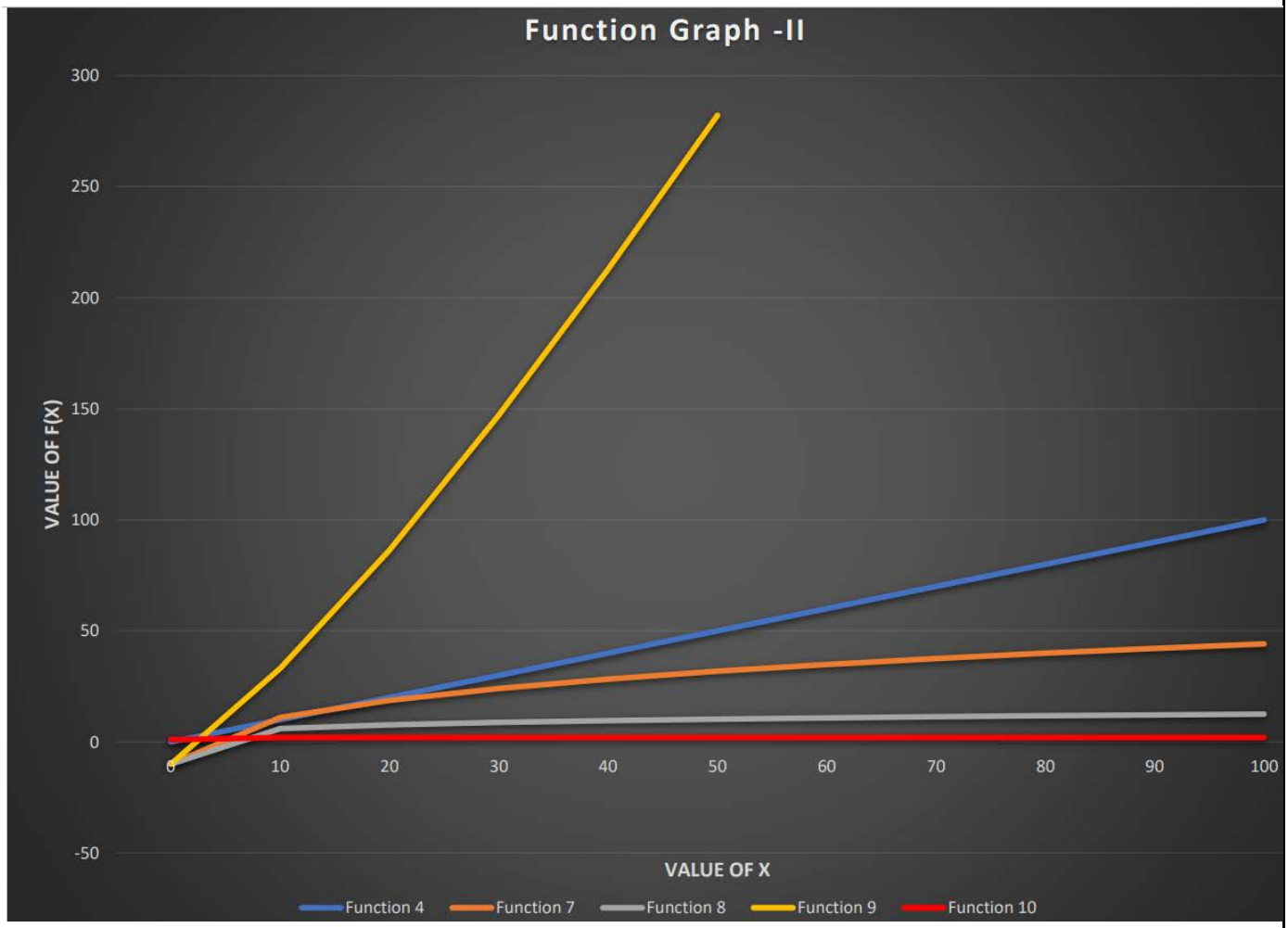
```c
#include<stdio.h>
#include<math.h>
double fact(int x){
    if(x==0)
      return 1;
    else
      return x*fact(x-1);
}
int f1(int x){
   return x;
}
double f2(int x){
   return log(x);
}
double f3(int x){
   return log2(log2(x));
}
double f4(int x){
   return pow(2,log2(x));
}
double f5(int x){
   return log2(x);
}
double f6(int x){
   return pow(sqrt(2),log2(x));
}
double f7(int x){
   return pow(log2(x),2);
}
```

```
double f8(int x){
    return pow(2,sqrt(2*log2(x)));
}
double f9(int x){
    return x*log2(x);
}
double f10(int x){
    return pow(x,1.0/log2(x));
}
int main(){
    printf("\nX\tF1\tF2\tF3\tF4\tF5\tF6\tF7\tF8\tF9\tF10");
    for(int i=0;i<=100;i+=10){
        printf("\n%d\t",i);
        printf("%d\t",f1(i));
        printf("%.2lf\t",f2(i));
        printf("%.2lf\t",f3(i));
        printf("%.2lf\t",f4(i));
        printf("%.2lf\t",f5(i));
        printf("%0.2lf\t",f6(i));
        printf("%.2lf\t",f7(i));
        printf("%.2lf\t",f8(i));
        printf("%.2lf\t",f9(i));
        printf("%.2lf",f10(i));
    }
    printf("\n\nX\tFactorial Function");
    for(int i=0;i<=20;i+=2){
        printf("\n%d\t%.2lf",i,fact(i));
    }
}
```

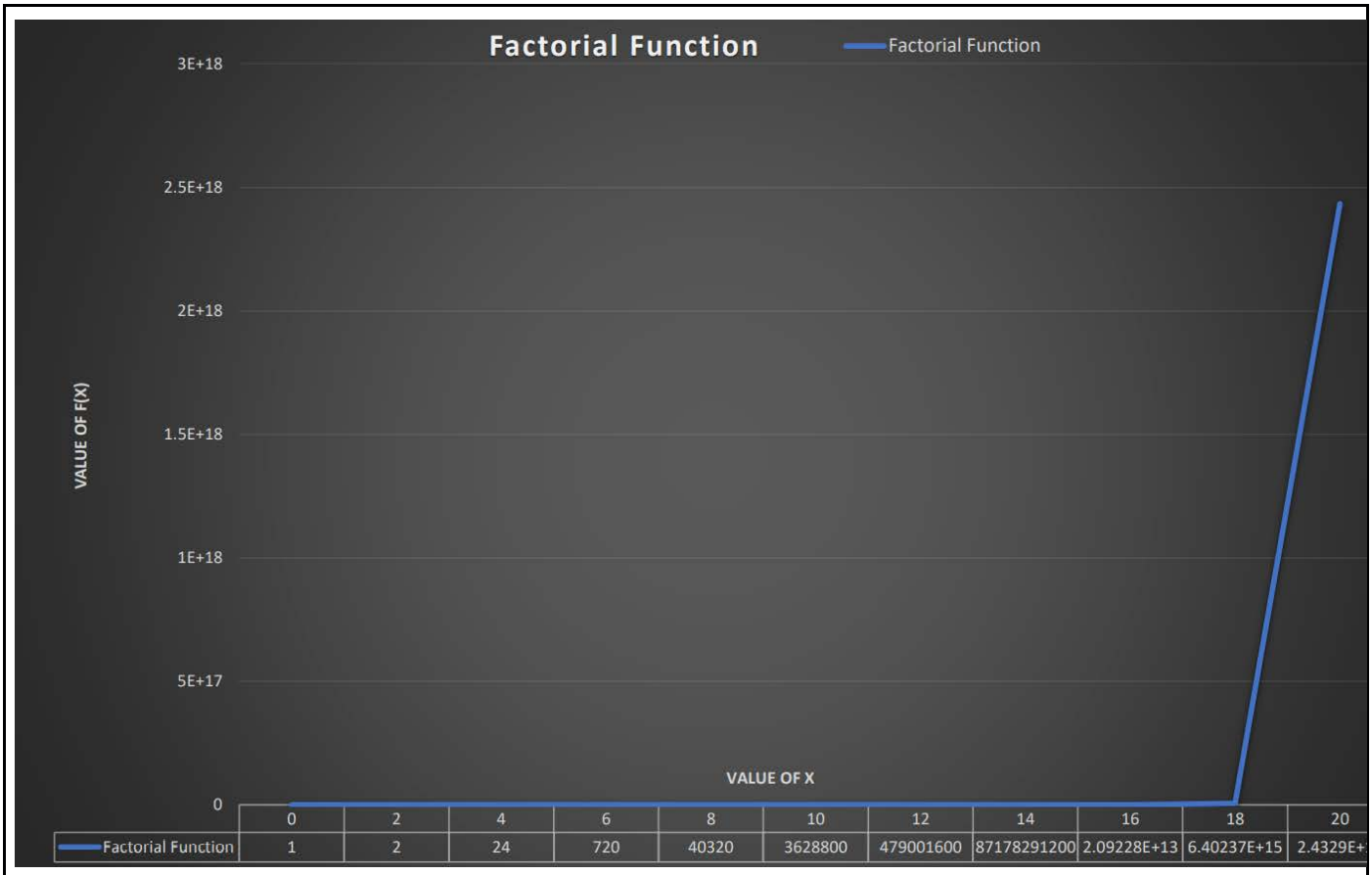
RESULT: The result of the following functions are shown below:-



From the above graph we observe that the Function 1 gives a linear graph that is y is proportional to x while the Functions 2,3,5 and 6 are increasing in a logarithmic manner. The increase of values in the Function 1 is constant and is greater than rest of the functions. The functions 2,3 and 5 produces an indeterminate value at x=0 which here is shown with the help of negative Y.



From the above graph we observe that Function 4 is directly proportional to x that is linearly increasing with x. The functions 7 and 8 are increasing in a logarithmic form. The function 9 is also increasing in a logarithmic manner but the value of F(x) for function 9 far exceeds the other functions. The function 10 gives a value 1 at x = 0 and then it is constant for the rest of the values of x. The functions 7,8 and 9 produces an indeterminate value at x=0 which here is shown with the help of negative Y.



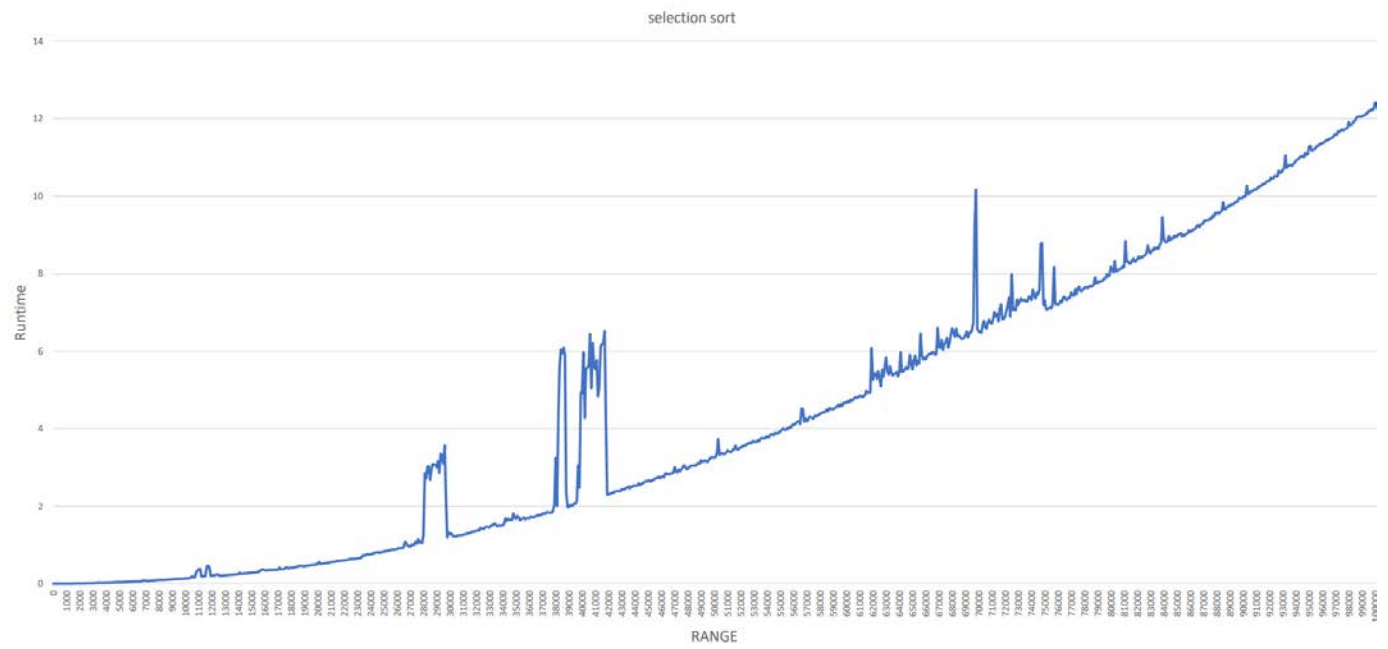
In the above graph we can observe that the increase in the graph is in an irregular form and the graph will reach to an infinite value at some x.

PART 1-B Program 1	
PROBLEM STATEMENT :	Each student have to generate random 100000 numbers using rand() function and use this input as 1000 blocks of 100 integer numbers to Selection sorting algorithm.
ALGORITHM/ THEORY:	Theory:- It first finds the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array. In this algorithm, the array is divided into two parts, first is sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and unsorted part is the given array. Sorted part is placed at the left, while the unsorted part is placed at the right. In selection sort, the first smallest element is selected from the unsorted array and placed at the first position. After that second smallest element is selected and placed in the second position. The process continues until the array is entirely sorted. Algorithm:- Use the rand() function to collect 100000 random numbers and store in the random numbers in the a text file. Step 1: Initialize minimum value index(min_val) from the set of numbers to index 0. Step 2: Traverse the array to find the minimum value. Step 3: If we find any element smaller than the min_val then swap the values. Step 4: Update the min_val to point to the next element. Step 5: Keep repeating the process till all the numbers are sorted.

PROGRAM:

```
#include<stdio.h>
#include<time.h>
int min(int arr[],int k,int size){
    int pos = k;
    for(int i = k+1;i<size;i++){
        if(arr[i]<arr[pos]){
            pos = i;
        }
    }
    return pos;
}
void selection_sort(int arr[],int size){
    int temp,m,i;
    for(i=0;i<size;i++){
        m = min(arr,i,size);
        temp = arr[i];
        arr[i] = arr[m];
        arr[m] = temp;
    }
}
int main(){
    int arr[100000],arr2[100000],y=0;
    clock_t t1,t2;
    FILE *f;
    f = fopen("num.txt","r");
    for(int i = 0;i<100000;i++){
        fscanf(f,"%d",&arr[i]);
        arr2[i] = arr[i];
    }
    fclose(f);
    while(y < 100000){
        y +=100;
        t1 = clock();
        selection_sort(arr,y);
        t2 = clock();
        double total = ((double)(t2 - t1))/CLOCKS_PER_SEC;
        printf("0 - %d\t%lf \n",y-1,total);
        for(int i=0;i<y;i++){
            arr[i] = arr2[i];
        }
    }
}
```

RESULT:

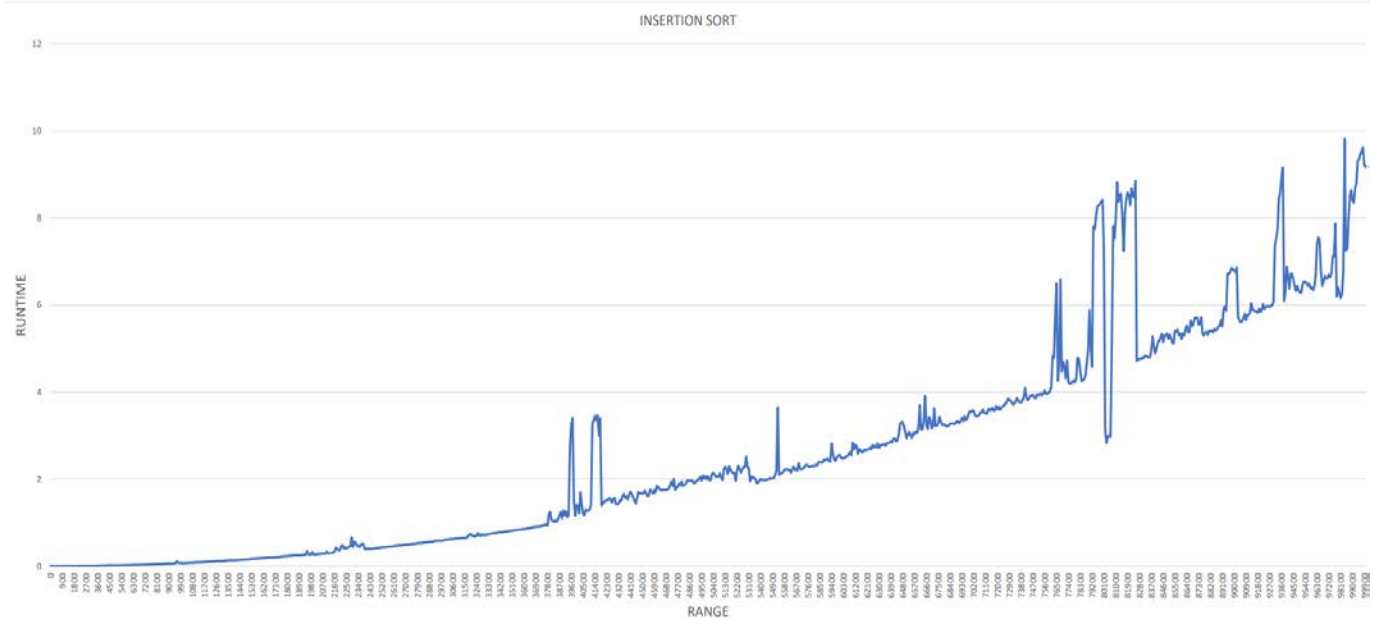


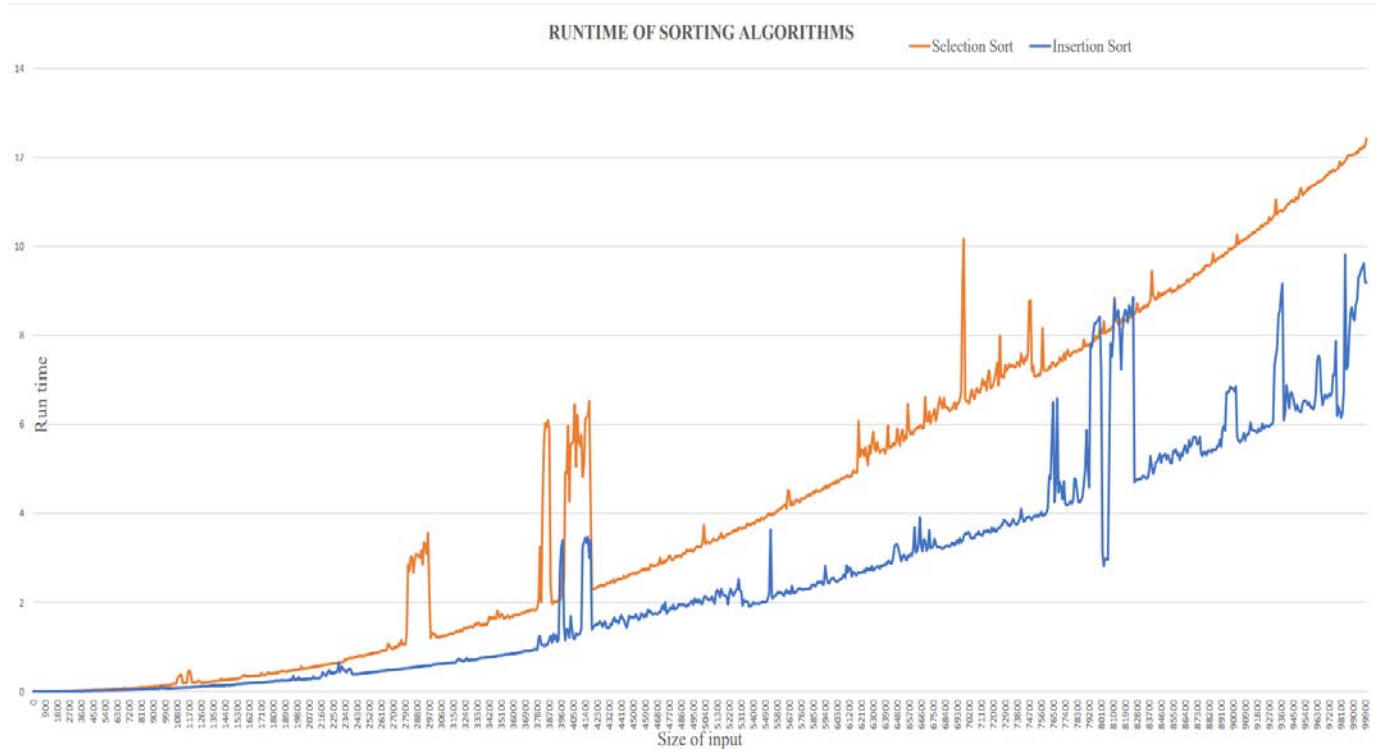
PART 1-B Program 2	
PROBLEM STATEMENT:	Each student have to generate random 100000 numbers using rand() function and use this input as 1000 blocks of 100 integer numbers to insertion sorting algorithm.
ALGORITHM/ THEORY:	Theory:- It works similar to the sorting of playing cards in hands. It is assumed that the first card is already sorted in the card game, and then we select an unsorted card. If the selected unsorted card is greater than the first card, it will be placed at the right side; otherwise, it will be placed at the left side. Similarly, all unsorted cards are taken and put in their exact place. Algorithm:- Use the rand() function to collect 100000 random numbers and store in the random numbers in the a text file. Step 1: Traverse the array from position 0 to n-1. Step 2: Compare the key element with its predecessor. Step 3: If the key element is smaller than its predecessor, compare it with the elements before. Copy the greater elements one position up to make place for the key. Step 4: Keep repeating the process for all the elements.

PROGRAM:	`#include<stdio.h>`
	`#include<time.h>`
	`void insertion_sort(int arr[],int y){`
	`  int num,j;`
	`  for(int i = 1;i<y;i++){`
	`  num = arr[i];`
	`  j = i-1;`
	`  while((num<arr[j]) && j>=0){`
	`  arr[j+1] = arr[j];`
	`  j--;`
	`  }`
	`  arr[j+1] = num;`
	`  }`

```
}
int main(){
    int arr[100000],arr2[100000],y=0;
    clock_t t1,t2;
    FILE *f;
    f = fopen("num.txt","r");
    for(int i = 0;i<100000;i++){
      fscanf(f,"%d",&arr[i]);
      arr2[i] = arr[i];
    }
    fclose(f);
    while(y < 100000){
        y +=100;
        t1 = clock();
        insertion_sort(arr,y);
        t2 = clock();
        double total = ((double)(t2 - t1))/CLOCKS_PER_SEC;
        printf("0 - %d\t%lf \n",y-1,total);
        for(int i=0;i<y;i++){
            arr[i] = arr2[i];
        }
    }
}
```

RESULT:





From the above picture we observe that the selection sort algorithm has a higher runtime then the insertion sort algorithm. The runtimes shown here in the graph may differ from as the type of processor also affects the runtime. To sum it all insertion sort is better than selection sort as it takes less time to execute.

CONCLUSION:	We saw various Functions and analyzed their graphs and drew our conclusions based on the graphs in the observation part above. Apart from that we saw two sorting algorithms Insertion and selection sort and the time they take to sort a large input here 100000. We saw that selection sort is a bit slower as compared to insertion sort.