Experiment No.	**1 B**
Aim	**Experiment on finding the running time of an algorithm**
Name	**Rucha Sudhir Kulkarni**
UID No.	**2021300067**
Class & Division	**SE Computer Engineering (Div:A)(Batch:D)**

Aim:

Each student have to generate random 100000 numbers using rand() function and use this input as 1000 blocks of 100 integer numbers to Insertion and Selection sorting algorithms.

Algorithm:

1. Start
2. Create 2 integer arrays of length 100000.
3. Input 100000 random integers into both the arrays using : rand()%10000
4. Store the generated random numbers in a text file.
5. Perform insertion sort and selection sort of all the elements in groups of 100, then 200, then 300 ..so on till end.
6. Print the time taken for each sorting using clock() function.
7. Stop
8. Void insertionsort(int arr[],int n):
   - **8.1.** Consider first element to be already sorted
   - **8.2.** Pick the next element, and store it separately in a **key.**
   - **8.3.** Now, compare the **key** with all elements in the sorted array.
   - **8.4.** If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
   - **8.5.** Insert the value.
   - **8.6.** Repeat until the array is sorted.
9. Void selectionsort(int arr[],int n):
   - **9.1.** Initialize minimum value(min_idx) to location 0.
   - **9.2.** Traverse the array to find the minimum element in the array.
   - **9.3.** While traversing if any element smaller than min_idx is found then swap both the values.
   - **9.4.** Then, increment min_idx to point to the next element.
   - **9.5.** Repeat until the array is sorted.

Program:

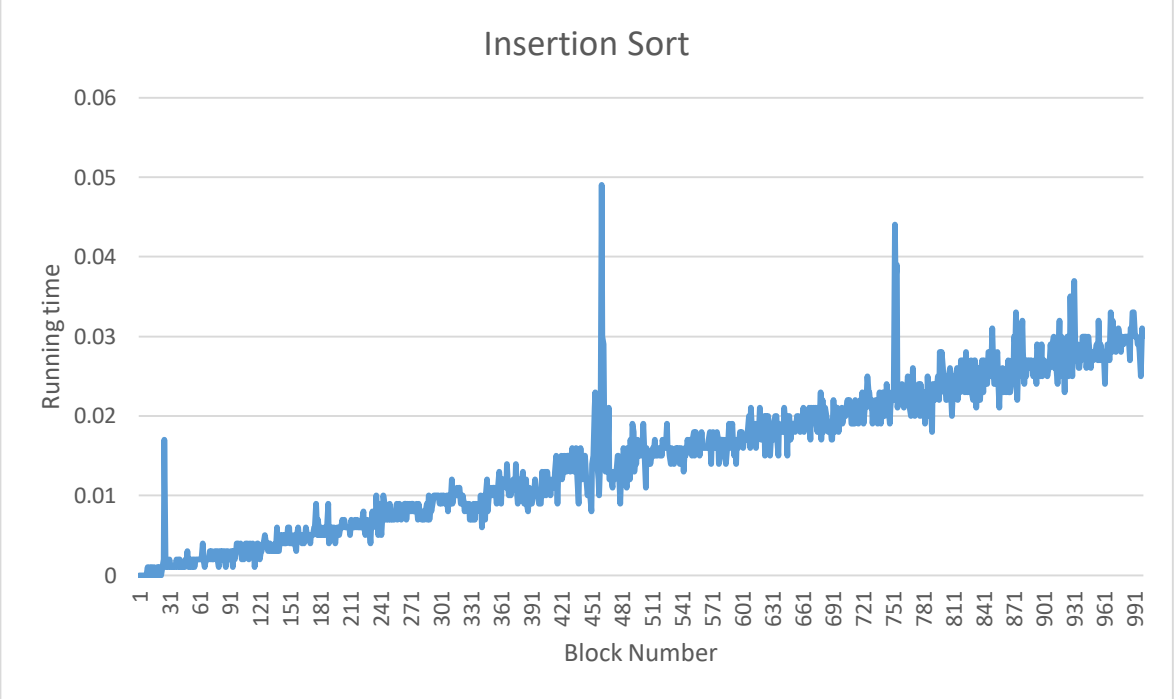
```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
```

```c
void insertionsort(int arr[],int n){

for(int i=1;i<n;i++){
int ele=arr[i];
int j=i-1;

while(j>=0 && ele<=arr[j]){//checking the elements before index i
arr[j+1]=arr[j];//shifting to right
j--;
}
arr[j+1]=ele;
}


}
void swap(int *a, int *b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}

void selectionSort(int array[], int size) {
  for (int step = 0; step < size - 1; step++) {
    int min_idx = step;
    for (int i = step + 1; i < size; i++) {

      // To sort in descending order, change > to < in this line.
      // Select the minimum element in each loop.
      if (array[i] < array[min_idx])
        min_idx = i;
    }

    // put min at the correct position
    swap(&array[min_idx], &array[step]);
  }
}
int main(){
FILE *fptr;
FILE *ansdoc;
FILE *timedoc;
FILE *ansdoc2;
FILE *timedoc2;
clock_t start, end;

fptr = fopen("demo.txt","w");
ansdoc = fopen("ans.txt","w");
timedoc = fopen("time.txt","w");
ansdoc2 = fopen("ans2.txt","w");
timedoc2 = fopen("time2.txt","w");
```

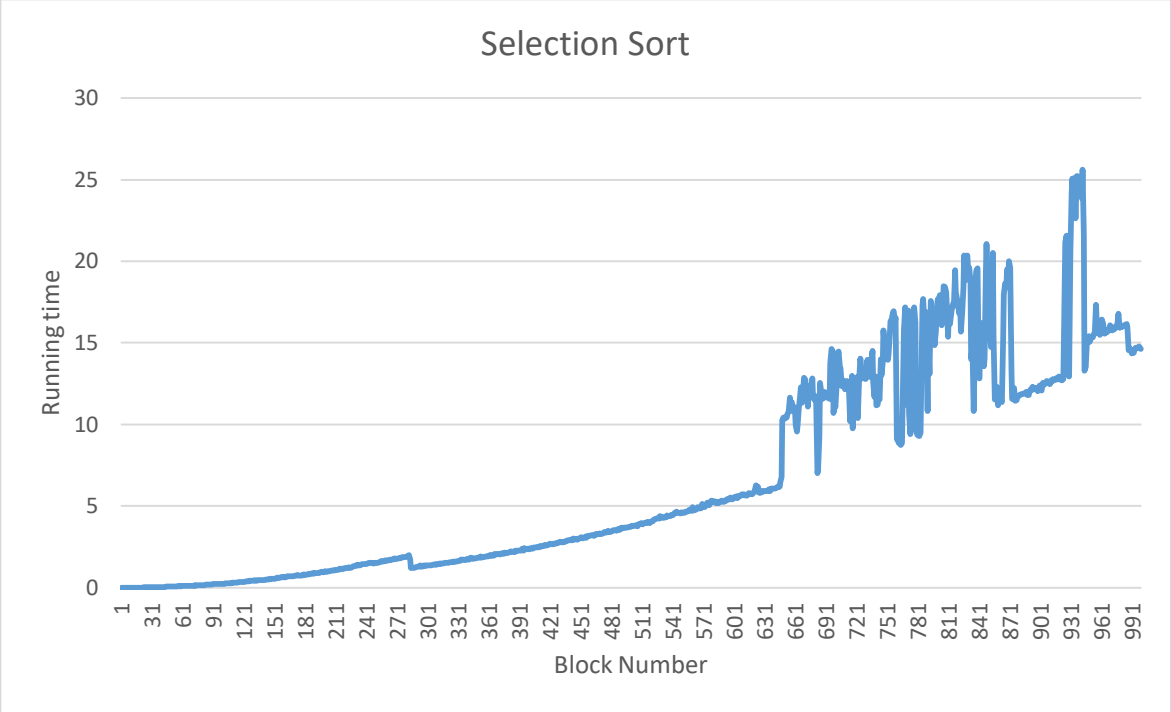
```
/*if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }*/
int arr[100000],arr2[100000];
for(int i=0;i<100000;i++){
int x=rand()%10000;
printf("%d\n",x);
fprintf(fptr,"%d\n",x);
arr[i]=x;
arr2[i]=x;
}
// for(int i=1;i<=1000;i++)
// {
//     start = clock();

// insertionsort(arr,i*100);
// end = clock();


// double time_taken = (double)(end - start) / (double)(CLOCKS_PER_SEC);
// printf("Time taken for %d elements to sort using insertion sort:%f
s  \n",i*100,time_taken);
// fprintf(timedoc,"%f\n",time_taken);
// }

for(int i=1;i<=1000;i++)
{
    start = clock();

selectionSort(arr2,i*100);
end = clock();


double time_taken = (double)(end - start) / (double)(CLOCKS_PER_SEC);
printf("Time taken for %d elements to sort using selection sort:%f
s  \n",i*100,time_taken);
fprintf(timedoc2,"%f\n",time_taken);
}


// for(int i=1;i<=1000;i++){
//     //printf("\nSorted arr %d\n",i);
//     fprintf(ansdoc,"Sorted arr %d\n",i);
//     for(int j=0;j<i*100;j++){
// //printf("%d\n",arr[j]);
// fprintf(ansdoc,"%d\n",arr[j]);
// }
```

```
// }

for(int i=1;i<=1000;i++){
    //printf("\nSorted arr %d\n",i);
    fprintf(ansdoc2,"Sorted arr %d\n",i);
    for(int j=0;j<i*100;j++){
//printf("%d\n",arr[j]);
fprintf(ansdoc2,"%d\n",arr2[j]);
}
}
fclose(fptr);
fclose(ansdoc);
fclose(timedoc);
fclose(ansdoc2);
fclose(timedoc2);
return 0;
}
```

Graph & observation:



The graph is approximately increasing which means that the running time is increasing with increase in number of elements to be sorted using insertion sort.



Initially the increase in running time with increase in number of elements is less, then there is quite variation and ups and downs in the running time for selection sort.

In general the running time of insertion sort is less than selection sort.
Space complexity of both algorithms is almost same because both use the same variables and memory.

Conclusion:

After performing the above experiment, I got to know how to find running time of any function in C using the clock() function. Also I learnt two sorting algorithms, namely insertion sort and selection sort.