NAME:	Yash Ravindra Burad
UID:	2022301004
SUBJECT	DAA
EXPERIMENT NO :	1B
AIM:	Experiment on finding the running time of an algorithm.
OBJECTIVE:	To find out running time of 2 sorting algorithms like Selection sort and Insertion sort.
ALGORITHM	Main function: step 1: start Step2: call generate_numbers() function Step 2: call operation()function Step 3: end generate_numbers() function: step 1: start step 2: crate the file pointer step 3: open the file in writing mode step 3: starts the loop from 0 to 100000 step 4: insert the 100000 random numbers in the file step 5: close the file handle step 6: end operation function(): step 1: start step 2: open the file in reading mode step 3: start the loop from 0 to 100000 and increment it with 100 step 4: create two arrays step 5: start the loop from 0 to j and scan the data from file

step 6: before sorting store the time
step 7: perform selection sort
step 8: check the time after after the sorting
step 9: calculate the time taken by the algorithm
step 10: before sorting store the time
step 11: perform selection sort
step 12: check the time after after the sorting
step 13: calculate the time taken by the algorithm

selection sort:
step 1: start
step 2: start the loop
step 3: initialize the min element
step 4: start the loop from i+1 to n
step 5: check the condition:
        if jth element less than min element then minimum
element will be j.
step 6: if minimum element not equal to i,
then initialize variable t with array(i)
perform ith element = array of min
array(min) = t
step 7: end.

Insertion sort:
Step 1: start
Step 2: start the loop from 1 to n
Step 3: initialize j with i-1
Step 4: current element is array(i)
Step 5: if array(key)>0 and j>=0
        Repeat below steps 6,7
Step 6: j+1th element will jth element
Step 7: decrement j
Step 8: array(j+1) = current.
Step 9: end.

| PROGRAM: | ```c
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
void selectionsort(int arr[],int n)
{
    for(int i=0;i<n;i++)
    {
        int min_ind=i;
        for(int j=i+1;j<n;j++)
        {
            if(arr[j]<arr[min_ind])
            min_ind=j;
        }
        if(min_ind!=i)
        {
            int t=arr[i];
            arr[i]=arr[min_ind];
        arr[min_ind]=t;
        }
    }
}
void insertionsort(int arr[],int n)
{
    for(int i=1;i<n;i++)
    {
        int j=i-1;
        int key=arr[i];
        while(j>=0 && arr[j]>key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void generate_numbers()
{

  FILE *ptr;
  ptr=fopen("number.txt","w");
  for(int i=0;i<100000;i++)
  {
``` |

```c
        fprintf(ptr,"%d\n",rand() % 100000);
    }
    fclose(ptr);
}
void operation()
{
    FILE *ptr;
    ptr=fopen("number.txt","r");
    for(int j=0;j<100000;j+=100)
    {
        int arr1[j];
        int arr2[j];
        for(int i=0;i<j;i++)
        {
            fscanf(ptr,"%d\n",&arr1[i]);
        }
        for(int i=0;i<j;i++)
        {
            arr2[i]=arr1[i];
        }
        clock_t start_selection=clock();
        selectionsort(arr1,j);
        clock_t end_selection = clock();
        double currs=(double)(end_selection-
start_selection)/CLOCKS_PER_SEC;


        clock_t start_insertion=clock();
        insertionsort(arr2,j);
        clock_t end_insertion=clock();
        double curri=(double)(end_insertion-
start_insertion)/CLOCKS_PER_SEC;
        printf("\n%d\t%f\t%f",j,currs,curri);
    }
}
int main()
{
    generate_numbers();
    operation();
    return 0;
}
```

Observation:
Graph for taken by Selection sort and Insertion sort:
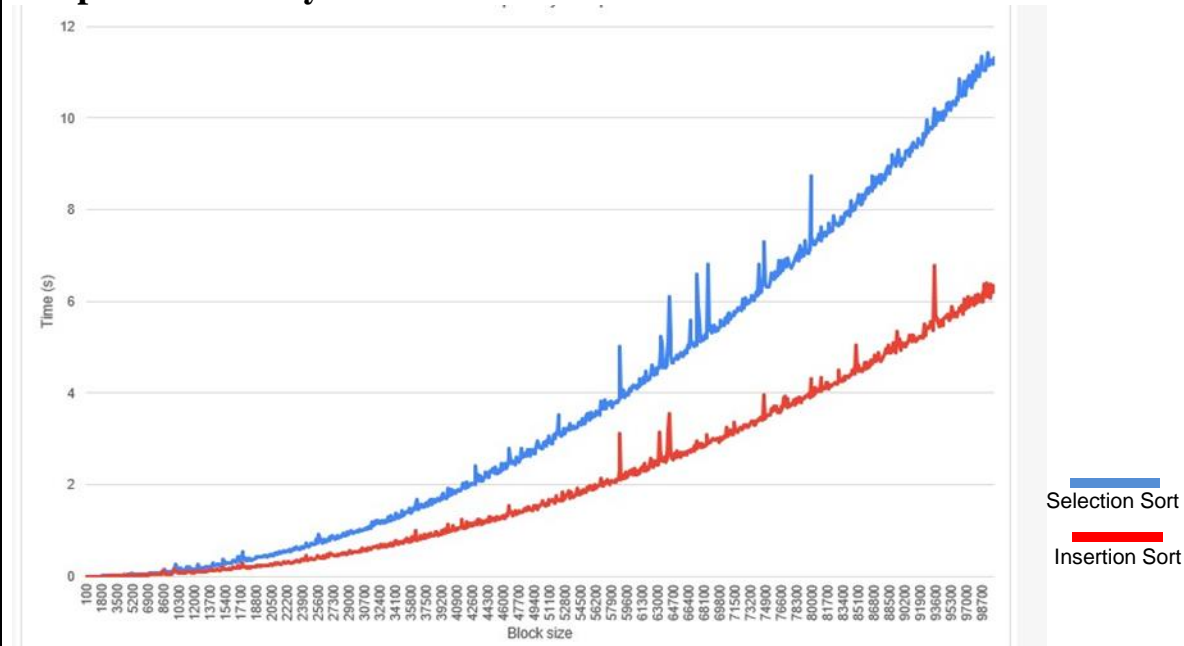
Observation:
As shown in the Graph, selection sort's graph increases above the Insertion sort graph. Selection sort takes more time to execute than Insertion sort. So**, Insertion sort is more fast, efficient than selection sort.**

CONCLUSION:	In this Practical, I learnt about Selection sort, insertion sort and found the running time for each sorting algorithm. I worked on random numbers and stored them into blocks and performed sorting on each block and calculated the time time take by each block to sort the random data.