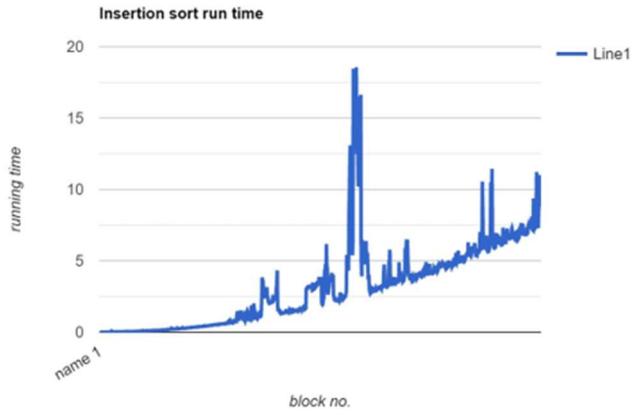
NAME:	Atul Sharma
UID:	2021700060
SUBJECT	DAA
EXPERIMENT NO :	1-B
AIM:	**Experiment on finding the running time of an algorithm.**
ALGORITHM	1.Start 2.initialize array a with size=100000 3for i=0 to i<1000 4.call function getData(i+1,a) 5.initialize start and end values of clock() function 6.call function insertionSort(a,(i+1)*100) 7.print array 8. call function getData(i+1,a) 9.initialize start and end values of clock() function 10.call function selectionSort(a,(i+1)*100) 11.print array 12.end for insertionSort() : n=length(A) for i=1 to n-1 do j=i while j>0 and A[j-1]>A[j] do swap (A[j],A[j-1]) j=j-1 end while end for

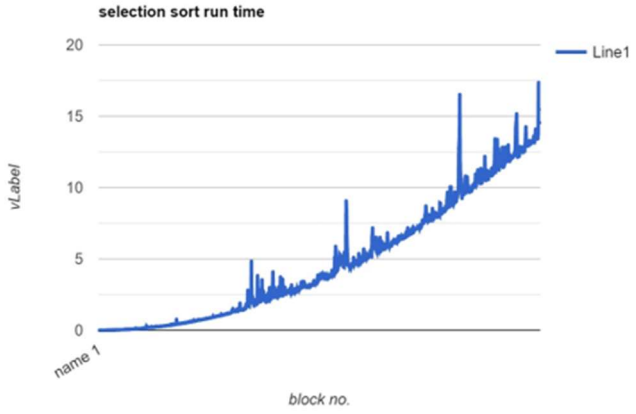
	procedure selection sort list : array of items n : size of list for i = 1 to n - 1 /* set current element as minimum*/ min = i /* check the element to be minimum */ for j = i+1 to n if list[j] < list[min] then min = j; end if end for /* swap the minimum element with the current element*/ if indexMin != i then swap list[min] and list[i] end if end for end procedure
PROGRAM:	```c #include <stdio.h> #include <math.h> #include <conio.h> #include <stdlib.h> #include <time.h> void getInput() { FILE *fp; fp = fopen("input.text","w"); ```

```
  for(int i=0;i<100000;i++)
  fprintf(fp,"%d ",rand()%100000);
  fclose(fp);
}


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


void selectionSort(int arr[], int len){
    int minIndex, temp;
    for(int i=0; i<len; i++){
        minIndex = i;
        for(int j=i+1; j<len; j++){
            if(arr[j] < arr[minIndex]){
                minIndex = j;
            }
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}


int main(){

    getInput();
    FILE *fp, *Wptr;
    int index=99;
    int arrNums[100000];
    clock_t t;
    fp = fopen("input.text", "r");
    Wptr = fopen("iTimes.txt", "w");
```

```
    for(int i=0; i<300; i++){
        for(int j=0; j<=index; j++){
            fscanf(fp, "%d", &arrNums[j]);
        }
        t = clock();
        insertionSort(arrNums, index+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        fprintf(Wptr, "time taken for %d iteration is
%Lf\n", (i+1), time_taken);
        printf("%d\t%lf\n", (i+1), time_taken);
        index = index + 100;
        fseek(fp, 0, SEEK_SET);
    }
    fclose(Wptr);
    Wptr = fopen("STimes.txt", "w");
    index=99;
    for(int i=0; i<300; i++){
        for(int j=0; j<=index; j++){
            fscanf(fp, "%d", &arrNums[j]);
        }
        t = clock();
        selectionSort(arrNums, index+1);
        t = clock() - t;
        double time_taken = ((double)t)/CLOCKS_PER_SEC;
        fprintf(Wptr, "time taken for %d iteration is
%Lf\n", (i+1), time_taken);
        printf("%d\t%lf\n", (i+1), time_taken);
        index = index + 100;
        fseek(fp, 0, SEEK_SET);
    }
    fclose(Wptr);
    fclose(fp);
    return 0;
}
```

RESULT (SNAPSHOT)





Time complexity:
Insertion sort: Best case-O(n)
Worst case-O(n^2)

Selection sort: Best and worst case- O(n^2)

CONCLUSION:	Through this experiment, I understood the concept of time complexity and as we increase the number of inputs the program take more time. Also through the analysis I came to know that for larger value of input number it is better to use insertion sort instead of selection sort.