Name	OM CHANDRA
UID no.	2021700014
Experiment No.	1-B

AIM:	Experiment on finding the running time of an algorithm.
Program 1	
PROBLEM STATEMENT :	For this experiment, you need to implement two sorting algorithms namely Insertion and Selection sort methods. Compare these algorithms based on time and space complexity. Time required to sorting algorithms can be performed using high_resolution_clock::now() under namespace std::chrono. You have togenerate1,00,000 integer numbers using C/C++ Rand function and save them in a text file. Both the sorting algorithms uses these 1,00,000 integer numbers as input as follows. Each sorting algorithm sorts a block of 100 integers numbers with array indexes numbers A[0..99], A[0..199], A[0..299],..., A[0..99999]. You need to use high_resolution_clock::now() function to find the time required for 100, 200, 300.... 100000 integer numbers. Finally, compare two algorithms namely Insertion and Selection by plotting the time required to sort 100000 integers using LibreOffice Calc/MS Excel. The x-axis of 2-D plot represents the block no. of 1000 blocks. The y-axis of 2-D plot Represents the running time to sort 1000 blocks of 100,200,300,...,100000 integer numbers. Note – You have to use C/C++ file processing functions for reading and writing randomly generated 100000 integer numbers.
ALGORITHM/ THEORY:	**Insertion sort Algorithm:** • Iterate from arr[1] to arr[N] over the array. • Compare the current element (key) to its predecessor. • If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element. **Selection sort Algorithm:** • Initialize minimum value(**min_idx**) to location 0.

	• Traverse the array to find the minimum element in the array. • While traversing if any element smaller than **min_idx** is found then swap both the values. • Then, increment **min_idx** to point to the next element. • Repeat until the array is sorted. 1: Start. 2: Include the required libraries stdio.h, stdlib.h, time.h, and limits.h. 3: Define two sorting functions as per problem statemebt selection_sort and insertion_sort. 4: In the main function, using file handling open the file for writing. 5: Generate 1000 blocks of 100 random numbers each and store them in the file. 6: Close the file after writing. 7: Open the file for reading. 8: For each block of 100 elements, read the elements from the file into two arrays. 9: Sort the elements of array using the selection_sort function. 10: Use clock() to measure the time taken by the algorithm, and store the value inside a variable. 11: Sort the elements of array using the insertion_sort function. 12: Use clock() to measure the time taken by the algorithm, and store the value inside a variable. 13: Display the number of blocks and time taken by both of the algorithm to sort a specific blocks. 14: Repeat the process for 750 blocks. 15: Close the file after reading. 16: Stop.

PROGRAM:	#include<stdio.h>
	#include<stdlib.h>
	#include<time.h>
	#include<limits.h>
	
	void selection_sort(int arr[],int size) {
	for(int i=0;i<size-1;i++) {
	int min=i;
	for(int j=i+1;j<size;j++)
	if(arr[j]<arr[min])
	min = j;
	int temp = arr[min];
	arr[min] = arr[i];
	arr[i] = temp;
	}
	
	}

```
void insertion_sort(int arr[],int n) {
   int i,key,j;
   for(int 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 main() {
   FILE *filep;
   filep  = fopen ("exp1b.txt", "w");
   srand((unsigned int) time(NULL));
   for(int block=0;block<1000;block++) {
      for(int i=0;i<100;i++) {
         int number = (int)(((float) rand() / (float)(RAND_MAX))*100000);
         fprintf(filep,"%d ",number);
      }
      fputs("\n",filep);
   }
   fclose (filep);
   filep = fopen("exp1b.txt", "r");
  printf("Block\tSelection Sort Time(ms)\tInsertion Sort Time(ms)\n");
  for(int block=0;block<1000;block++) {
   clock_t t1,t2;
   int arr[(block+1)*100];
   int arr1[(block+1)*100];
   for(int i=0;i<(block+1)*100;i++){
      fscanf(filep, "%d", &arr[i]);
      arr1[i] = arr[i];
   }
   fseek(filep, 0, SEEK_SET);
   t1 = clock();
   selection_sort(arr,(block+1)*100);
   t1 = clock() - t1;
```

```
    t2 = clock();
    insertion_sort(arr1,(block+1)*100);
    t2 = clock() - t2;

    double selection_sort_time = ((double)t1)/CLOCKS_PER_SEC;
    double insertion_sort_time = ((double)t2)/CLOCKS_PER_SEC;

    printf("%d\t%f\t\t\t%f\n",(block+1),selection_sort_time,insertion_sort_time);
   }
    fclose(filep);
}
```

RESULT:

```
ine-Out-trzkphbl.kc2' '--stderr=Microsoft-MIEngine-Error-vezbp1yn.poc' '--pid=Microso
Block   Selection Sort Time(ms) Insertion Sort Time(ms)
1       0.000000                0.000000
2       0.000000                0.000000
3       0.000000                0.000000
4       0.000000                0.000000
5       0.001000                0.000000
6       0.000000                0.001000
7       0.000000                0.001000
8       0.001000                0.000000
9       0.001000                0.001000
10      0.001000                0.000000
11      0.001000                0.001000
12      0.002000                0.001000
13      0.001000                0.001000
14      0.002000                0.001000
15      0.003000                0.001000
16      0.003000                0.001000
17      0.003000                0.002000
18      0.003000                0.002000
19      0.004000                0.002000
20      0.004000                0.003000
21      0.005000                0.003000
22      0.005000                0.003000
23      0.005000                0.004000
24      0.006000                0.003000
25      0.007000                0.003000
26      0.007000                0.004000
27      0.008000                0.005000
28      0.008000                0.005000
29      0.008000                0.005000
30      0.010000                0.005000
31      0.009000                0.006000
32      0.010000                0.006000
33      0.011000                0.007000
34      0.013000                0.009000
35      0.013000                0.007000
36      0.015000                0.008000
37      0.013000                0.008000
38      0.015000                0.009000
39      0.015000                0.010000
40      0.016000                0.011000
41      0.017000                0.010000
42      0.018000                0.011000
43      0.018000                0.011000
44      0.020000                0.011000
45      0.022000                0.004000
46      0.022000                0.002000
47      0.022000                0.001000
48      0.023000                0.001000
49      0.024000                0.001000
50      0.025000                0.001000
51      0.027000                0.002000
52      0.028000                0.001000
53      0.030000                0.002000
```

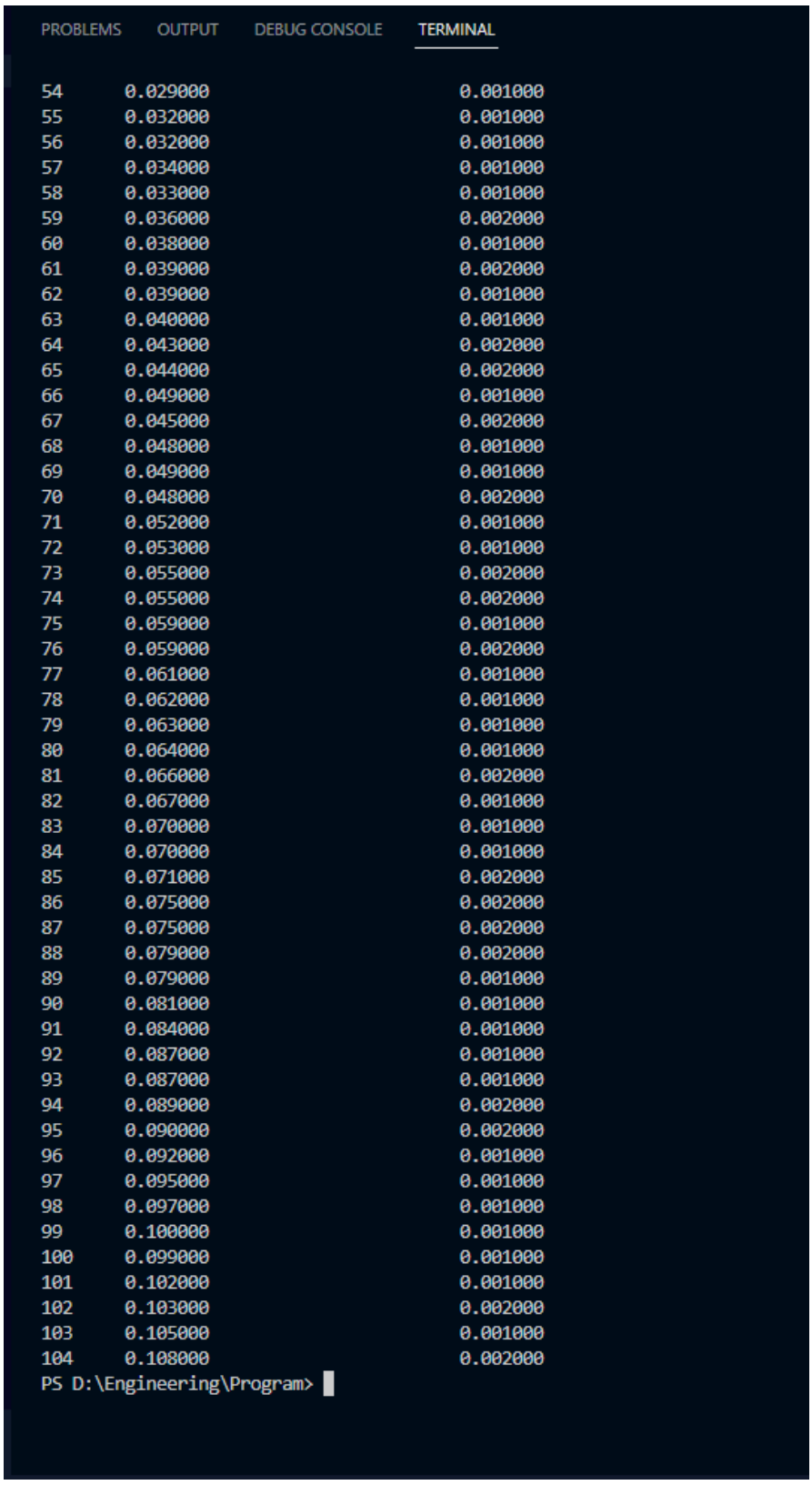
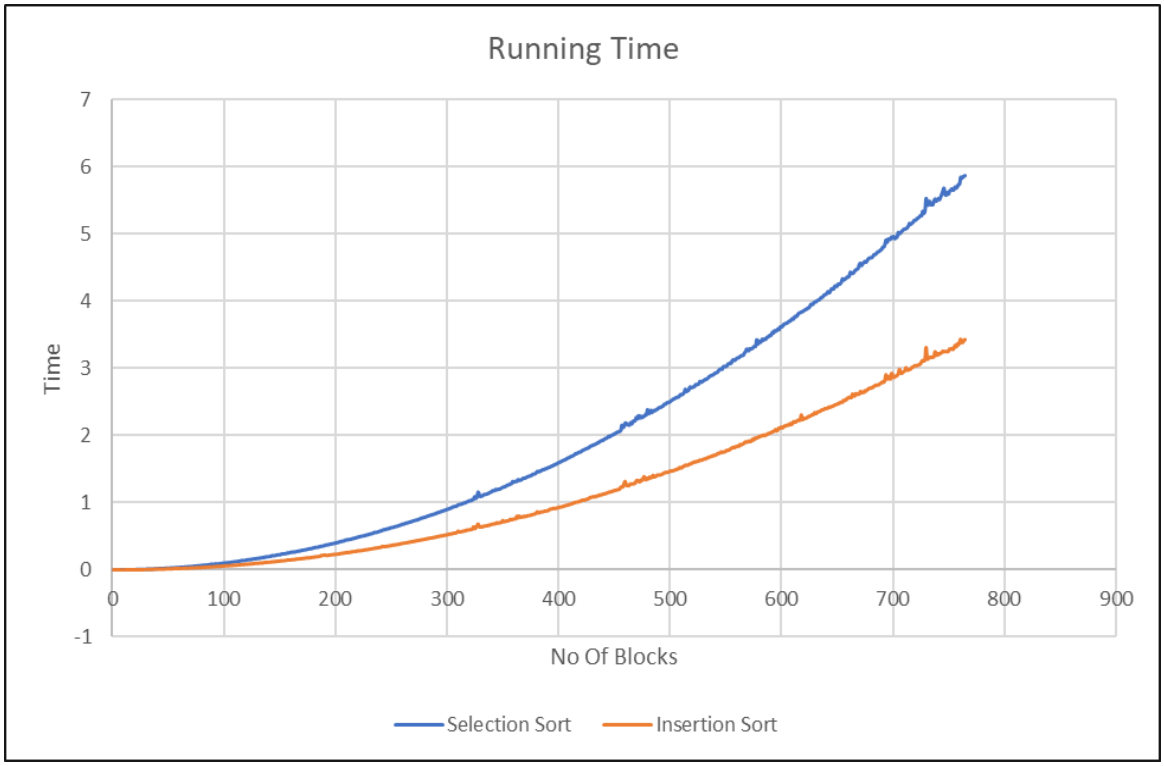

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
54 0.029000 0.001000
55 0.032000 0.001000
56 0.032000 0.001000
57 0.034000 0.001000
58 0.033000 0.001000
59 0.036000 0.002000
60 0.038000 0.001000
61 0.039000 0.002000
62 0.039000 0.001000
63 0.040000 0.001000
64 0.043000 0.002000
65 0.044000 0.002000
66 0.049000 0.001000
67 0.045000 0.002000
68 0.048000 0.001000
69 0.049000 0.001000
70 0.048000 0.002000
71 0.052000 0.001000
72 0.053000 0.001000
73 0.055000 0.002000
74 0.055000 0.002000
75 0.059000 0.001000
76 0.059000 0.002000
77 0.061000 0.001000
78 0.062000 0.001000
79 0.063000 0.001000
80 0.064000 0.001000
81 0.066000 0.002000
82 0.067000 0.001000
83 0.070000 0.001000
84 0.070000 0.001000
85 0.071000 0.002000
86 0.075000 0.002000
87 0.075000 0.002000
88 0.079000 0.002000
89 0.079000 0.001000
90 0.081000 0.001000
91 0.084000 0.001000
92 0.087000 0.001000
93 0.087000 0.001000
94 0.089000 0.002000
95 0.090000 0.002000
96 0.092000 0.001000
97 0.095000 0.001000
98 0.097000 0.001000
99 0.100000 0.001000
100 0.099000 0.001000
101 0.102000 0.001000
102 0.103000 0.002000
103 0.105000 0.001000
104 0.108000 0.002000
PS D:\Engineering\Program>

Chart:	
Observation:	We plotted the 2-D graph which represents the blocks and the time taken by both of the sorting algorithm(Selection sort and Insertion sort), X-axis represents no of blocks and Y-axis represents Time, in the graph it is clearly visible that the time taken to sort the blocks are more for selection sort than insertion sort.
CONCLUSION:	Thus, we have found the running time of insertion sort and selection sort on each block, and plotted a 2-D chart which shows the comparison of both algorithm's running time.