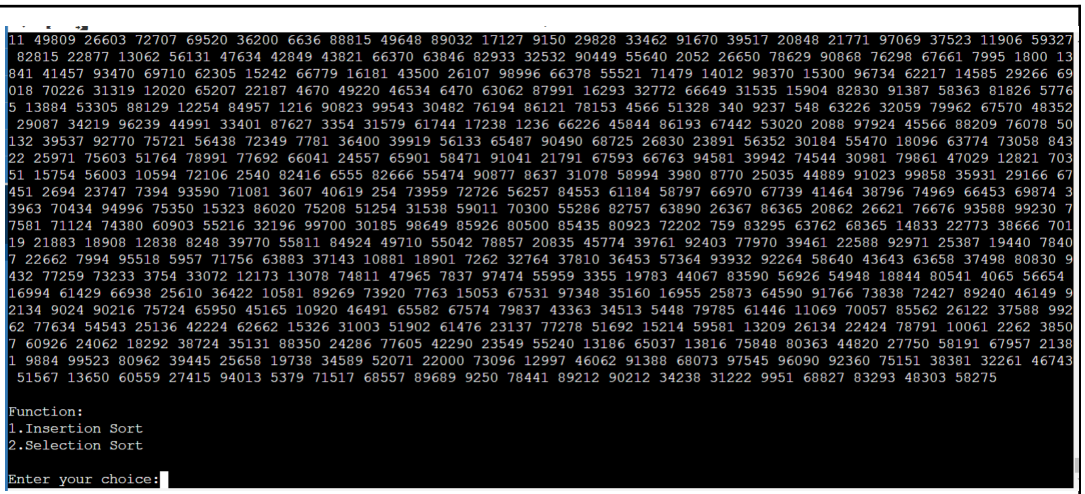
NAME:	Ayush Bodade
UID:	2021300015
SUBJECT:	DAA
EXPERIMENT NO:	02
DATE OF PERFORMANCE:	02/2/23
DATE OF SUBMISSION:	09/2/23
AIM:	Experiment on finding the running time of an algorithm.
Theory:	**Details**: The understanding of running time of algorithms is explored by implementing two basic sorting algorithms namely Insertion and Selection sorts. These algorithms work as follows. **Insertion sort:** 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. **Selection sort:** 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 the sorted part, and another one is the unsorted part. Initially, the sorted part of the array is empty, and the 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. **Problem Definition & Assumptions:** 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

	sorting algorithms can be performed using high_resolution_clock::now() under namespace std::chrono. You have to generate 1,00,000 integer numbers using the C/C++ Rand function and save them in a text file. Both the sorting algorithm uses these 1,00,000 integer numbers as input as follows. Each sorting algorithm sorts a block of 100 integer 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 the 2-D plot represents the block no. of 1000 blocks. The y-axis of the 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.

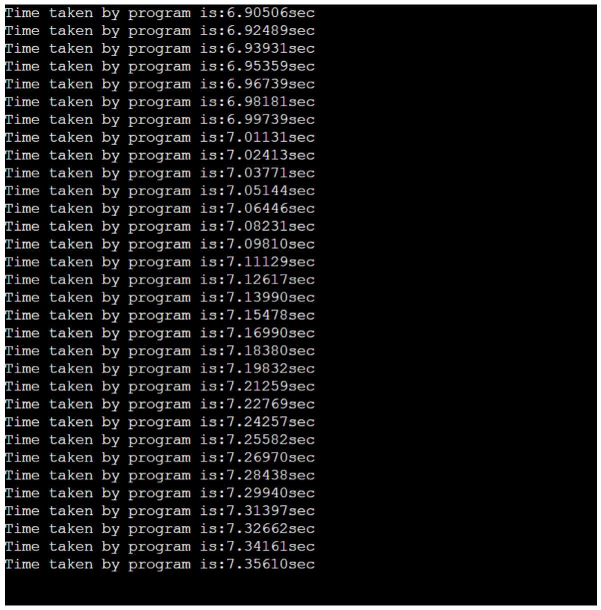
Algorithm:	Insertion Sort Function: ● Iterate from arr[1] to arr[N] over the array. ● Compare the current element (key) to its predecessor. ● If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element. Selection Sort Function: ● 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. GetInput Function: ● Function to make 100000 random numbers to sort and put into a text file Readfile Function: ● Function to read numbers from the text file Main Function: 1.Make a menu driven function and ask user his choice of sorting technique 2.If insertion sort, call the function and calculate the time interval at every 100 numbers getting sorted up to 1000 times/block. 3.If Selection sort, call the function and calculate the time interval at every 100 numbers getting sorted up to 1000 times/block. 4.Else if, invalid input. **Code link:** https://github.com/ayushbodade/daa_sem_4/blob/main/expt2/expt2.cpp

Results:



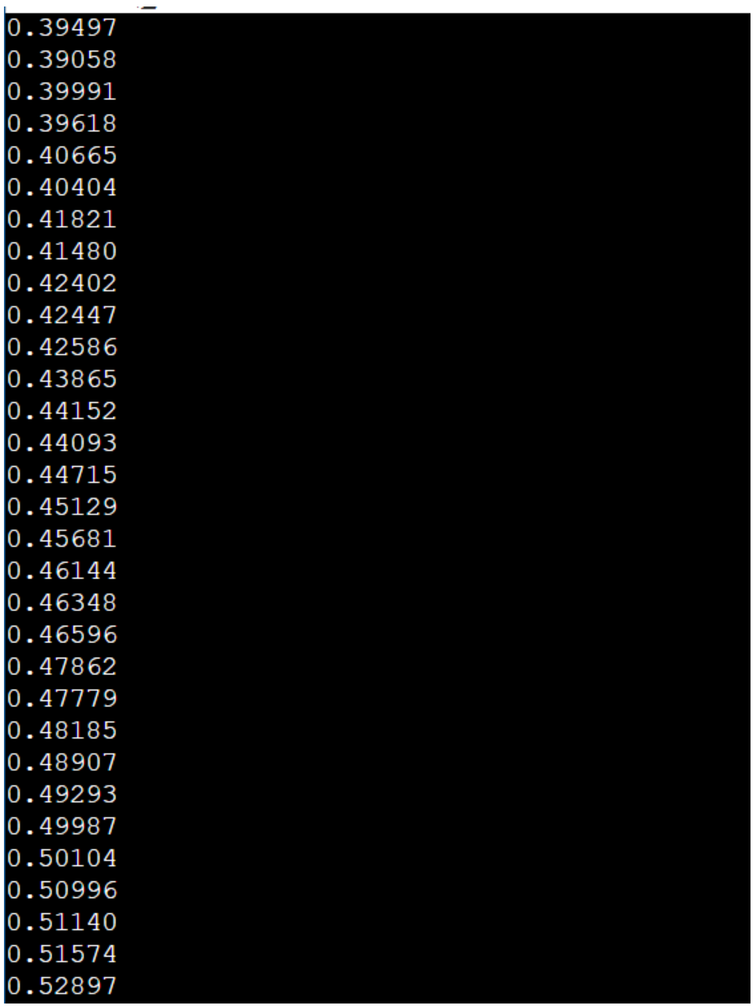
```
11 49809 26603 72707 69520 36200 6636 88815 49648 89032 17127 9150 29828 33462 91670 39517 20848 21771 97069 37523 11906 59327
 82815 22877 13062 56131 47634 42849 43821 66370 63846 82933 32532 90449 55640 2052 26650 78629 90868 76298 67661 7995 1800 13
841 41457 93470 69710 62305 15242 66779 16181 43500 26107 98996 66378 55521 71479 14012 98370 15300 96734 62217 14585 29266 69
018 70226 31319 12020 65207 22187 4670 49220 46534 6470 63062 87991 16293 32772 66649 31535 15904 82830 91387 58363 81826 5776
5 13884 53305 88129 12254 84957 1216 90823 99543 30482 76194 86121 78153 4566 51328 340 9237 548 63226 32059 79962 67570 48352
 29087 34219 96239 44991 33401 87627 3354 31579 61744 17238 1236 66226 45844 86193 67442 53020 2088 97924 45566 88209 76078 50
132 39537 92770 75721 56438 72349 7781 36400 39919 56133 65487 90490 68725 26830 23891 56352 30184 55470 18096 63774 73058 843
22 25971 75603 51764 78991 77692 66041 24557 65901 58471 91041 21791 67593 66763 94581 39942 74544 30981 79861 47029 12821 703
51 15754 56003 10594 72106 2540 82416 6555 82666 55474 90877 8637 31078 58994 3980 8770 25035 44889 91023 99858 35931 29166 67
451 2694 23747 7394 93590 71081 3607 40619 254 73959 72726 56257 84553 61184 58797 66970 67739 41464 38796 74969 66453 69874 3
3963 70434 94996 75350 15323 86020 75208 51254 31538 59011 70300 55286 82757 63890 26367 86365 20862 26621 76676 93588 99230 7
7581 71124 74380 60903 55216 32196 99700 30185 98649 85926 80500 85435 80923 72202 759 83295 63762 68365 14833 22773 38666 701
19 21883 18908 12838 8248 39770 55811 84924 49710 55042 78857 20835 45774 39761 92403 77970 39461 22588 92971 25387 19440 7840
7 22662 7994 95518 5957 71756 63883 37143 10881 18901 7262 32764 37810 36453 57364 93932 92264 58640 43643 63658 37498 80830 9
432 77259 73233 3754 33072 12173 13078 74811 47965 7837 97474 55959 3355 19783 44067 83590 56926 54948 18844 80541 4065 56654
16994 61429 66938 25610 36422 10581 89269 73920 7763 15053 67531 97348 35160 16955 25873 64590 91766 73838 72427 89240 46149 9
2134 9024 90216 75724 65950 45165 10920 46491 65582 67574 79837 43363 34513 5448 79785 61446 11069 70057 85562 26122 37588 992
62 77634 54543 25136 42224 62662 15326 31003 51902 61476 23137 77278 51692 15214 59581 13209 26134 22424 78791 10061 2262 3850
7 60926 24062 18292 38724 35131 88350 24286 77605 42290 23549 55240 13186 65037 13816 75848 80363 44820 27750 58191 67957 2138
1 9884 99523 80962 39445 25658 19738 34589 52071 22000 73096 12997 46062 91388 68073 97545 96090 92360 75151 38381 32261 46743
 51567 13650 60559 27415 94013 5379 71517 68557 89689 9250 78441 89212 90212 34238 31222 9951 68827 83293 48303 58275

Function:
1.Insertion Sort
2.Selection Sort

Enter your choice:
```

Insertion Sort:

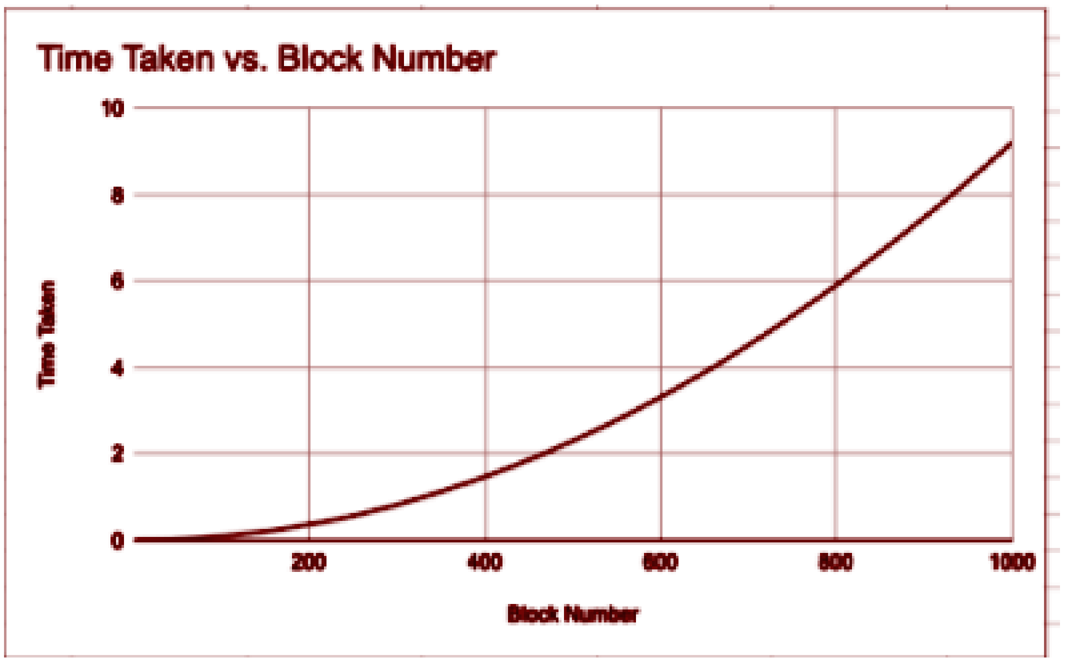
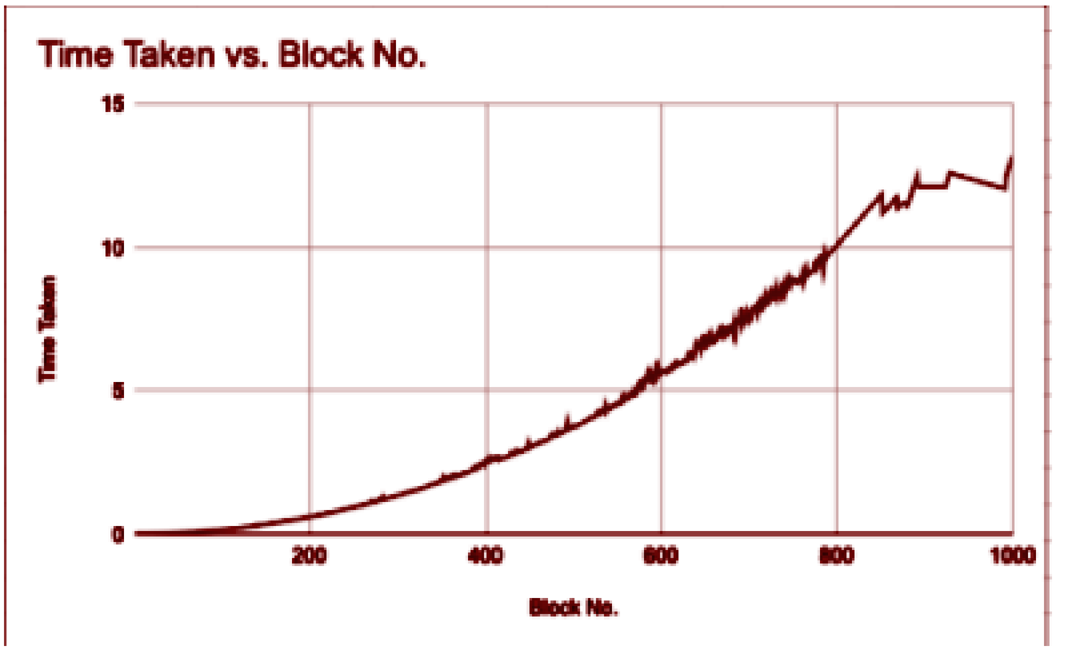


```
Time taken by program is:6.90506sec
Time taken by program is:6.92489sec
Time taken by program is:6.93931sec
Time taken by program is:6.95359sec
Time taken by program is:6.96739sec
Time taken by program is:6.98181sec
Time taken by program is:6.99739sec
Time taken by program is:7.01131sec
Time taken by program is:7.02413sec
Time taken by program is:7.03771sec
Time taken by program is:7.05144sec
Time taken by program is:7.06446sec
Time taken by program is:7.08231sec
Time taken by program is:7.09810sec
Time taken by program is:7.11129sec
Time taken by program is:7.12617sec
Time taken by program is:7.13990sec
Time taken by program is:7.15478sec
Time taken by program is:7.16990sec
Time taken by program is:7.18380sec
Time taken by program is:7.19832sec
Time taken by program is:7.21259sec
Time taken by program is:7.22769sec
Time taken by program is:7.24257sec
Time taken by program is:7.25582sec
Time taken by program is:7.26970sec
Time taken by program is:7.28438sec
Time taken by program is:7.29940sec
Time taken by program is:7.31397sec
Time taken by program is:7.32662sec
Time taken by program is:7.34161sec
Time taken by program is:7.35610sec
```

Selection Sort:

```
0.39497
0.39058
0.39991
0.39618
0.40665
0.40404
0.41821
0.41480
0.42402
0.42447
0.42586
0.43865
0.44152
0.44093
0.44715
0.45129
0.45681
0.46144
0.46348
0.46596
0.47862
0.47779
0.48185
0.48907
0.49293
0.49987
0.50104
0.50996
0.51140
0.51574
0.52897
```

Graph:	 
Conclusion:	Performing this experiment I understood the implementation of insertion sort and selection sort with their time complexity using graph plotting in MS excel