ACPCE

Where Knowledge is Second Nature

**Jawahar Education Societys Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai**

**NAME: PRIYUSH BHIMRAO KHOBRAGADE**

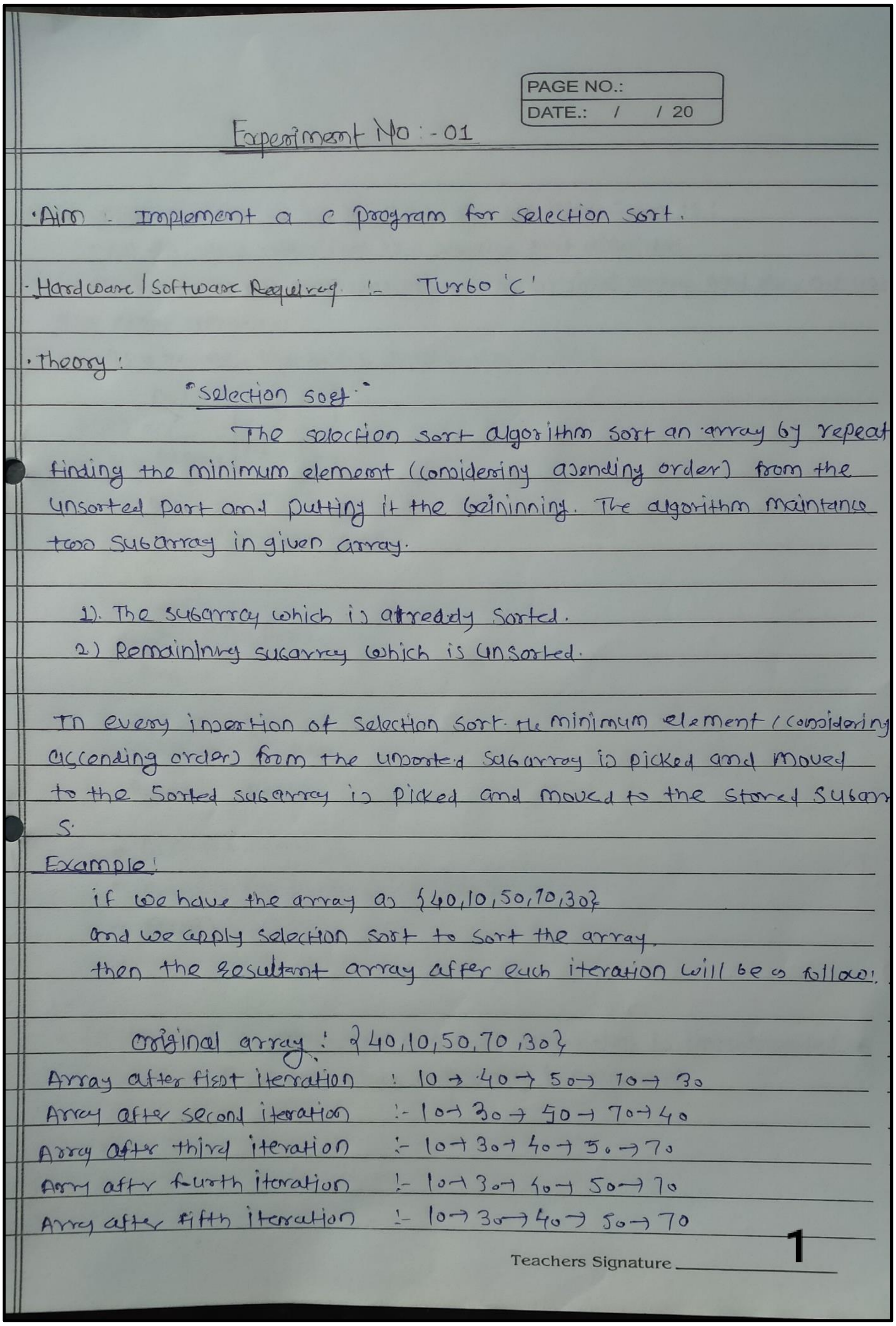
**PRN NO: 211112018**

**Roll No: 52**

**SUBJECT: Analysis of Algorithms Lab**

**Implement a C program for Selection sort**

**EXPERMINT: 01**



PAGE NO.:
DATE.: / / 20

Experiment No :- 01

·Aim : Implement a c program for selection sort.

·Hardware / Software Requireg :- Turbo 'C'

·Theory :

"Selection sort"

The selection sort algorithm sort an array by repeat finding the minimum element (considering asending order) from the unsorted part and putting it the beinninng. The algorithm maintance two subarray in given array.

1). The subarray which is already sorted.
2) Remaining subarray which is unsorted.

In every insertion of selection sort the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray is picked and moved to the stored subarr s.

Example:

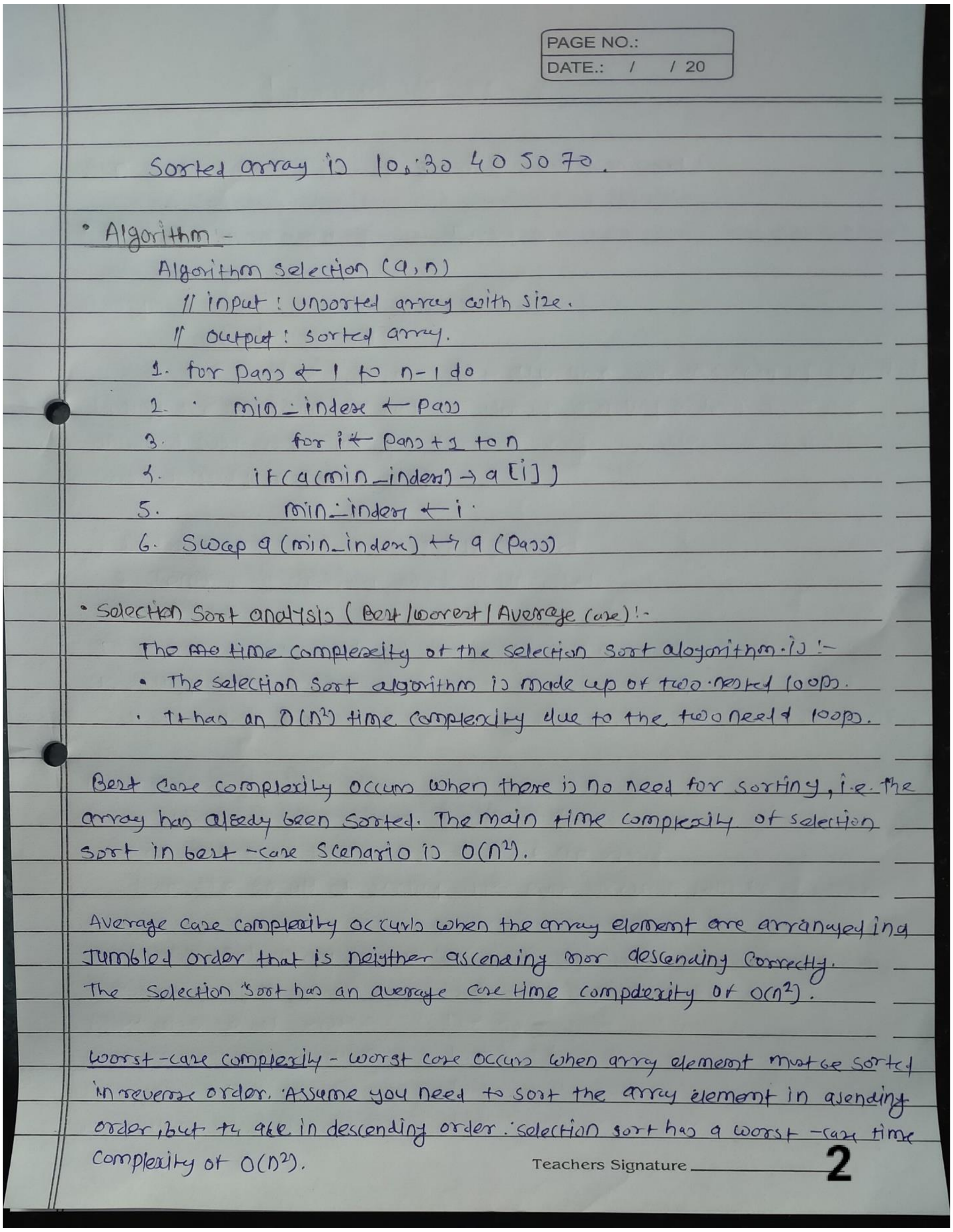
if we have the array as {40,10,50,70,30}
and we apply selection sort to sort the array.
then the resultant array after each iteration will be as follow:

Original array: {40,10,50,70,30}

| | |
|---|---|
| Array after first iteration | : 10 → 40 → 50 → 70 → 30 |
| Array after second iteration | :- 10 → 30 → 50 → 70 → 40 |
| Array after third iteration | :- 10 → 30 → 40 → 50 → 70 |
| Arry after furth iteration | :- 10 → 30 → 40 → 50 → 70 |
| Array after fifth iteration | :- 10 → 30 → 40 → 50 → 70 |

Teachers Signature ______

**Implement a C program for Selection sort**



PAGE NO.:
DATE.: / / 20

Sorted array is 10, 30 40 50 70.

• Algorithm -

```
Algorithm selection (a, n)
  // input : Unsorted array with size.
  // output : sorted array.
1. for pass ← 1 to n-1 do
2.    min_index ← pass
3.        for i ← pass + 1 to n
4.      if(a(min_index) > a [i] )
5.         min_index ← i
6. Swap a (min_index) ↔ a (pass)
```

• Selection Sort analysis ( Best / worst / Average case ) :-

The ~~Ao~~ time complexity of the selection sort alogorithm is :-

• The selection sort algorithm is made up of two nested loops.

• It has an $O(n^2)$ time complexity due to the two need loops.

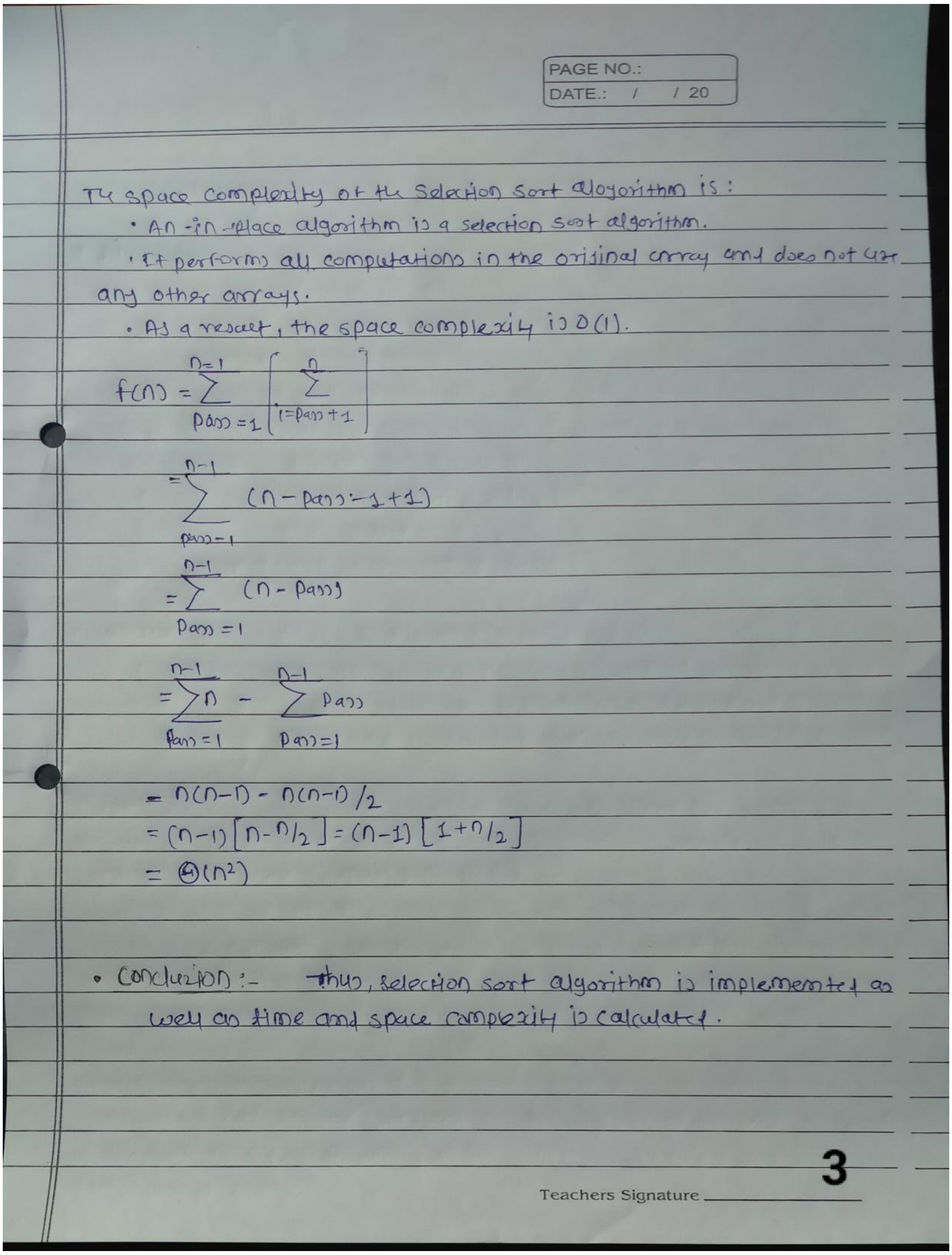
Best case complexity occurs when there is no need for sorting, i.e. the array has aleady been sorted. The main time complexity of selection sort in best-case Scenario is $O(n^2)$.

Average case complexity occurs when the array element are arranged in a jumbled order that is neiyther ascending nor descending correctly. The selection sort has an average case time compdexity of $O(n^2)$.

Worst-case complexity - worst case occurs when array element must be sorted in reverse order. Assume you need to sort the array element in asending order, but it are in descending order. selection sort has a worst-case time complexity of $O(n^2)$.

Teachers Signature

2



PAGE NO.:
DATE.: / / 20

The space complexity of the Selection sort algorithm is:

• An in-place algorithm is a selection sort algorithm.

• It performs all computations in the original array and does not use any other arrays.

• As a result, the space complexity is O(1).

$$f(n) = \sum_{pass=1}^{n=1} \left[ \sum_{i=pass+1}^{n} \right]$$

$$= \sum_{pass=1}^{n-1} (n - pass - 1 + 1)$$

$$= \sum_{pass=1}^{n-1} (n - pass)$$

$$= \sum_{pass=1}^{n-1} n - \sum_{pass=1}^{n-1} pass$$

$$= n(n-1) - n(n-1)/2$$

$$= (n-1)\left[n - n/2\right] = (n-1)\left[1 + n/2\right]$$

$$= \Theta(n^2)$$

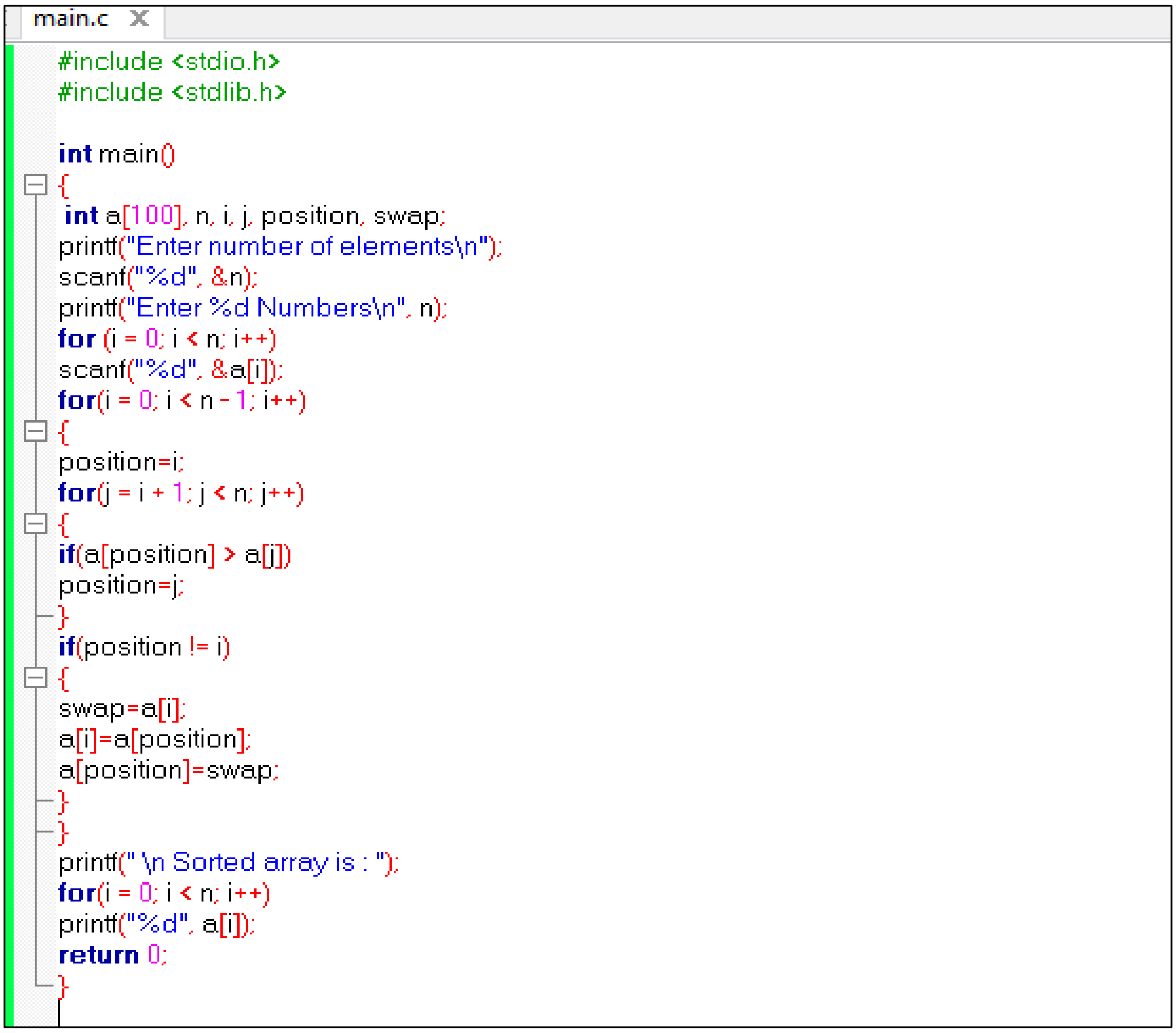
• Conclusion :- Thus, selection sort algorithm is implemented as well as time and space complexity is calculated.

3

Teachers Signature ____________

## Implement a C program for Selection sort

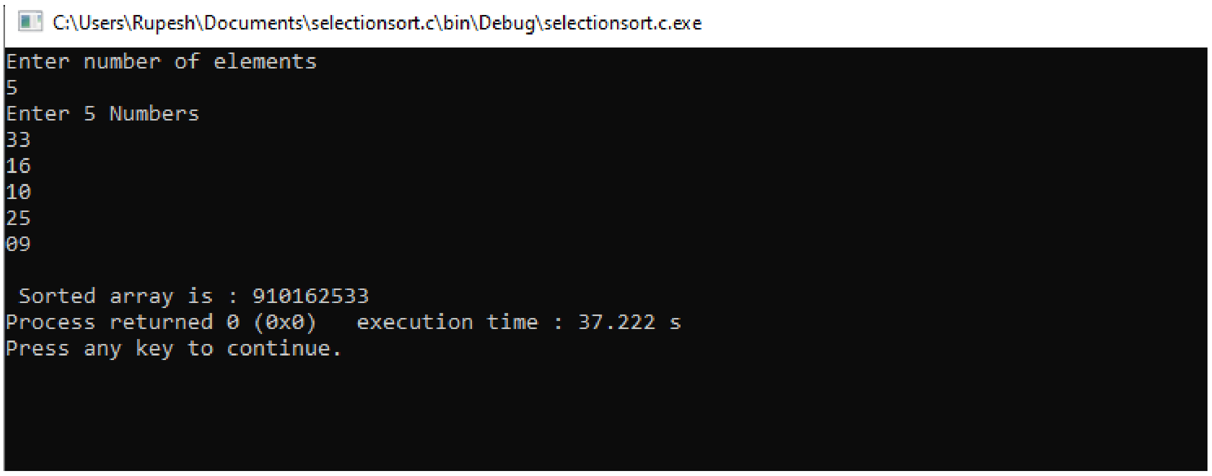
**Input:**



```
main.c ✕
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int a[100], n, i, j, position, swap;
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d Numbers\n", n);
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for(i = 0; i < n - 1; i++)
{
position=i;
for(j = i + 1; j < n; j++)
{
if(a[position] > a[j])
position=j;
}
if(position != i)
{
swap=a[i];
a[i]=a[position];
a[position]=swap;
}
}
printf(" \n Sorted array is : ");
for(i = 0; i < n; i++)
printf("%d", a[i]);
return 0;
}
```

**Output:**



```
C:\Users\Rupesh\Documents\selectionsort.c\bin\Debug\selectionsort.c.exe
Enter number of elements
5
Enter 5 Numbers
33
16
10
25
09

 Sorted array is : 910162533
Process returned 0 (0x0)   execution time : 37.222 s
Press any key to continue.
```

**Conclusion:** Thus, selection sort algorithm is implemented as well as time and space complexity is calculated.