Batch: B1 **Roll No.: 1711072**

Experiment No. 8

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Selection and Insertion Sort

Objective: Implementation of Selection & Insertion sort menu driven program.

Expected Outcome of Experiment:

CO	Outcome
CO3	Demonstrate sorting and searching methods

Books/ Journals/ Websites referred:

Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein; (CLRS)
"Introduction to Algorithms", Third Edition, The MIT Press.

Abstract:

DATA STRUCTURE:

1. int arr1[SIZE] //array of size SIZE=50 to store input elements for manipulation.
2. int arr2[SIZE] //array of size SIZE=50 to store input elements for manipulation.

OPERATIONS:

1. **void insertion()**

   Used for sorting array arr1 using insertion sort.

2. **void selection()**

   Used for sorting array arr2 using selection sort.

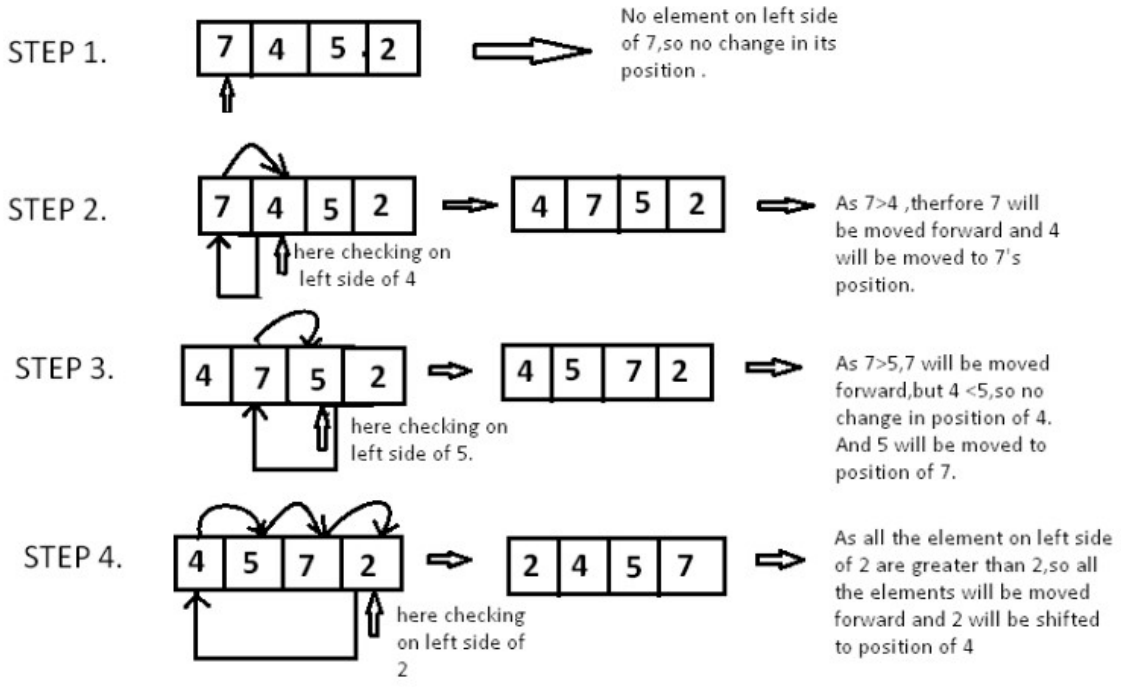
Related Theory: -

Insertion Sort:

- Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position i.e., the position to which it belongs in a sorted array.
- It iterates the input elements by growing the sorted array at each iteration. It compares the current element with the largest value in the sorted array.
- If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position.
- This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead.
- Time complexity:

  1. Best Case: $\Omega(n)$
  2. Average Case: $\theta(n^2)$
  3. Worst Case: $O(n^2)$

Take array $A[] = [7, 4, 5, 2]$.



Since 7 is the first element has no other element to be compared with, it remains at its position. Now when on moving towards 4, 7 is the largest element in the sorted list and greater than 4. So, move 4 to its correct position i.e. before 7. Similarly with 5, as 7 (largest element in the sorted list) is greater than 5, we will move 5 to its correct position. Finally for 2, all the elements on the left side of 2 (sorted list) are moved one position forward as all are greater than 2 and then 2 is placed in the first position. Finally, the given array will result in a sorted array.

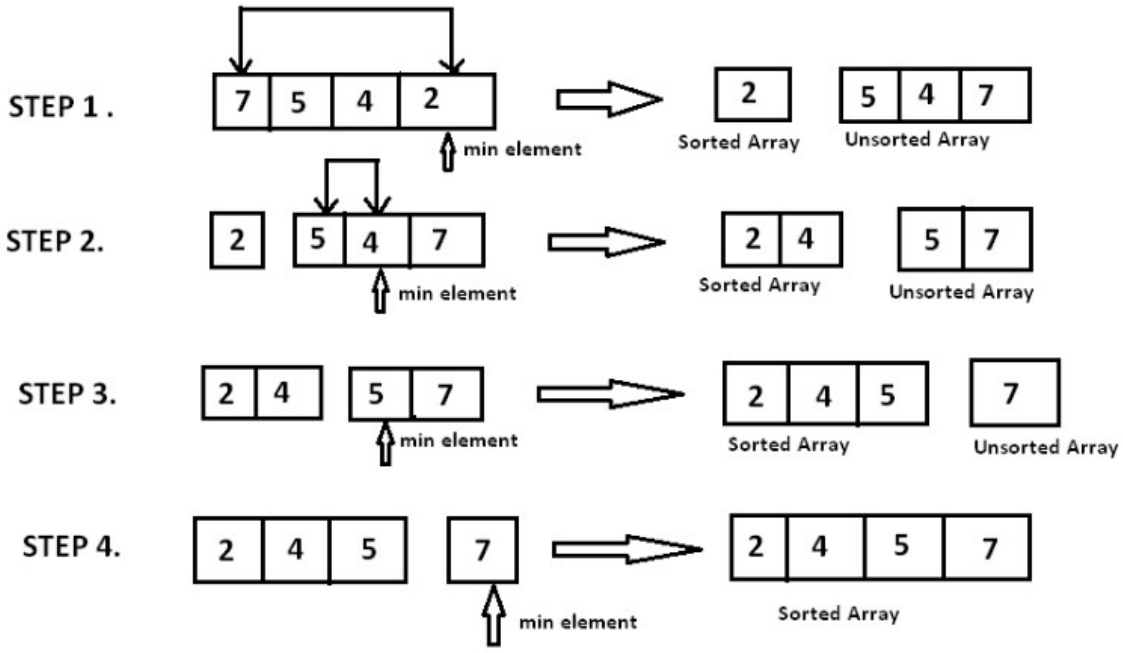
Selection Sort:

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

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

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

At i^{th} iteration, elements from position 0 to $i-1$ will be sorted.



Time complexity:

1. Best Case: $\Omega(n^2)$
2. Average Case: $\theta(n^2)$
3. Worst Case: $O(n^2)$

Implementation Details:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 50
int arr2[SIZE];
int arr1[SIZE];
int len;
void insertion(){
  int temp,i,j;
  for(i=1;i<len;i++)
  {
    temp=arr1[i];
    j=i-1;
    while(j>=0 && arr1[j]>temp){
      arr1[j+1]=arr1[j];
```

```
      j=j-1;
    }
    arr1[j+1]=temp;
    printf("\nAfter pass %d: ", i);
    for(int k=0;k<len;k++)
      printf("%d ", arr1[k]);
  }
}

void selection(){
  int temp, i,j,index;
  for(i=0;i<len-1;i++){
    index=i;
    for(j=i+1;j<len;j++){
      if(arr2[j]<arr2[index])
        index=j;
    }
    temp=arr2[index];
    arr2[index]=arr2[i];
    arr2[i]=temp;
    printf("\nAfter pass %d: ", i+1);
    for(int k=0;k<len;k++)
      printf("%d ", arr2[k]);
  }
}

int main(void) {
  int choice=10;
  printf("Enter number of elements you want to insert: ");
  scanf("%d", &len);
  printf("Enter your array elements:\n");
  for(int i=0;i<len;i++)
    scanf("%d", &arr1[i]);
  for(int i=0;i<len;i++)
    arr2[i]=arr1[i];
  while(choice!=3){
    printf("\n1. Insertion Sort\n2. Selection Sort\n3.
Exit\nEnter your choice: ");
    scanf("%d", &choice);
    switch(choice){
      case 1:
        insertion();
        break;
      case 2:
```

```
        selection();
        break;
      case 3:
        exit(1);
      default:
        printf("Enter a valid choice.");
    }
  }
}
```

For verification, my code is available on:
https://repl.it/@ARGHYADEEPDAS/InsertionAndSelectionSort

OUTPUT SCREEN:

```
Enter number of elements you want to insert:  5
Enter your array elements:
 20 32 10 90 5

1. Insertion Sort
2. Selection Sort
3. Exit
Enter your choice:  1

After pass 1: 20 32 10 90 5
After pass 2: 10 20 32 90 5
After pass 3: 10 20 32 90 5
After pass 4: 5 10 20 32 90
1. Insertion Sort
2. Selection Sort
3. Exit
Enter your choice:  2

After pass 1: 5 32 10 90 20
After pass 2: 5 10 32 90 20
After pass 3: 5 10 20 90 32
After pass 4: 5 10 20 32 90
1. Insertion Sort
2. Selection Sort
3. Exit
```

CONCLUSION:

The program ran successfully as we were able to implement menu driven program for insertion sort and selection sort and also show state of array after each pass.