Selection Sort

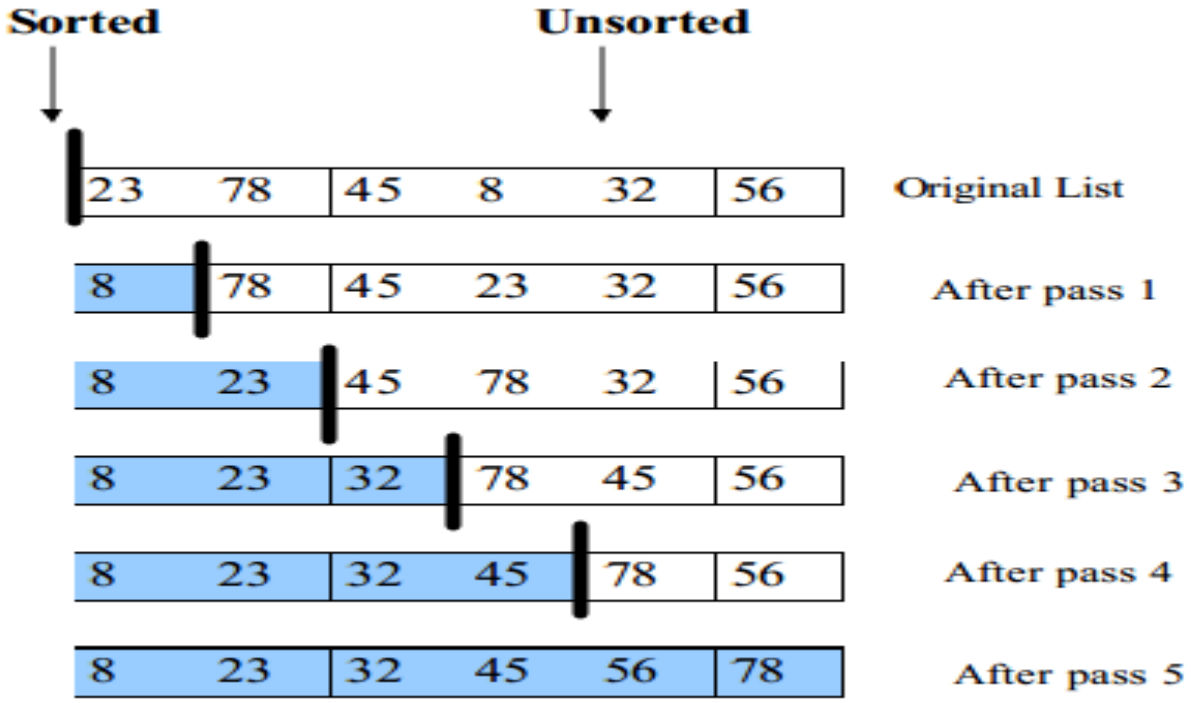
ð The list is divided into two sublists, sorted and unsorted, which are divided by an imaginary wall.

ð We find the smallest element from the unsorted sublist and swap it with the element at the beginning of the unsorted data.

ð After each selection and swapping, the imaginary wall between the two sublists move one element ahead, increasing the number of sorted elements and decreasing the number of unsorted ones.

ð Each time we move one element from the unsorted sublist to the sorted sublist, we say that we have completed a sort pass.

ð A list of n elements requires n-1 passes to completely rearrange the data



CODES(C)

```c
#include <stdio.h>

int main()
{
  int array[100], n, c, d, position, swap;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for ( c = 0 ; c < n ; c++ )
    scanf("%d", &array[c]);

  for ( c = 0 ; c < ( n - 1 ) ; c++ )
  {
    position = c;

    for ( d = c + 1 ; d < n ; d++ )
    {
      if ( array[position] > array[d] )
        position = d;
    }
    if ( position != c )
    {
      swap = array[c];
      array[c] = array[position];
      array[position] = swap;
    }
  }

  printf("Sorted list in ascending order:\n");

  for ( c = 0 ; c < n ; c++ )
    printf("%d\n", array[c]);

  return 0;
}
```

CODES(JAVA)

```
public class MySelectionSort {

    public static int[] doSelectionSort(int[] arr){

        for (int i = 0; i < arr.length - 1; i++)
        {
            int index = i;
            for (int j = i + 1; j < arr.length; j++)
                if (arr[j] < arr[index])
                    index = j;

            int smallerNumber = arr[index];
            arr[index] = arr[i];
            arr[i] = smallerNumber;
        }
        return arr;
    }

    public static void main(String a[]){

        int[] arr1 = {10,34,2,56,7,67,88,42};
        int[] arr2 = doSelectionSort(arr1);
        for(int i:arr2){
            System.out.print(i);
            System.out.print(", ");
        }
    }
}
```