PRACTICAL NO: 6

Name-Saloni Vishwakarma

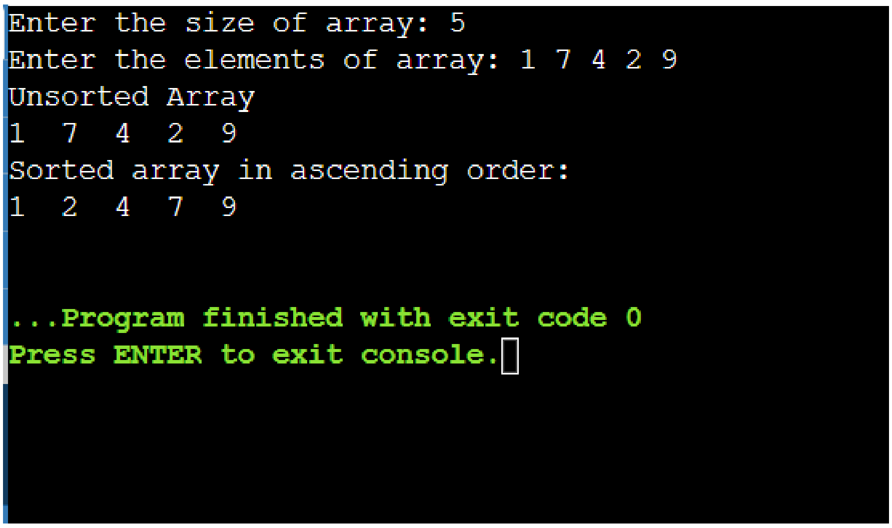
Batch-Roll no: C1-13

Aim: To study and implement different sorting algorithms.

Quick Sort:

```
#include <stdio.h>
// function to swap elements
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}

// function to find the partition position
int partition(int array[], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);

  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {

      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap the pivot element with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}

void quickSort(int array[], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on right of pivot
    int pi = partition(array, low, high);
    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

// function to print array elements
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

// main function
int main() {
  int n;
  printf("Enter the size of array: ");
  scanf("%d",&n);

  int data[n];
  printf("Enter the elements of array: ");
  for(int i=0;i<n;i++)
    scanf("%d",&data[i]);
  printf("Unsorted Array\n");
  printArray(data, n);

  // perform quicksort on data
  quickSort(data, 0, n - 1);

  printf("Sorted array in ascending order: \n");
  printArray(data, n);
}
```

Output:

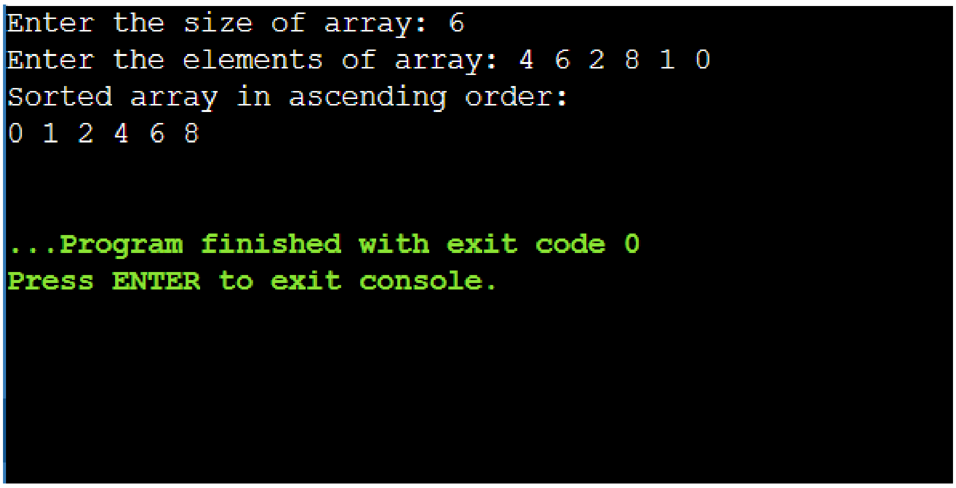


```
Enter the size of array: 5
Enter the elements of array: 1 7 4 2 9
Unsorted Array
1  7  4  2  9
Sorted array in ascending order:
1  2  4  7  9


...Program finished with exit code 0
Press ENTER to exit console.
```

Insertion Sort:

```
#include <stdio.h>

// Function to print an array
void printArray(int array[], int size) {
  for (int i = 0; i < size; i++) {
    printf("%d ", array[i]);
  }
  printf("\n");
}

void insertionSort(int array[], int size) {
  for (int step = 1; step < size; step++) {
    int key = array[step];
    int j = step - 1;

    // Compare key with each element on the left of it until an element smaller than
    // it is found.
    // For descending order, change key<array[j] to key>array[j].
    while (key < array[j] && j >= 0) {
      array[j + 1] = array[j];
      --j;
    }
    array[j + 1] = key;
  }
}

// Driver code
int main() {
  int n;
  printf("Enter the size of array: ");
  scanf("%d",&n);

  int data[n];
  printf("Enter the elements of array: ");
  for(int i=0;i<n;i++)
    scanf("%d",&data[i]);

  insertionSort(data, n);
  printf("Sorted array in ascending order:\n");
  printArray(data, n);
}
```

Output:



```
Enter the size of array: 6
Enter the elements of array: 4 6 2 8 1 0
Sorted array in ascending order:
0 1 2 4 6 8


...Program finished with exit code 0
Press ENTER to exit console.
```

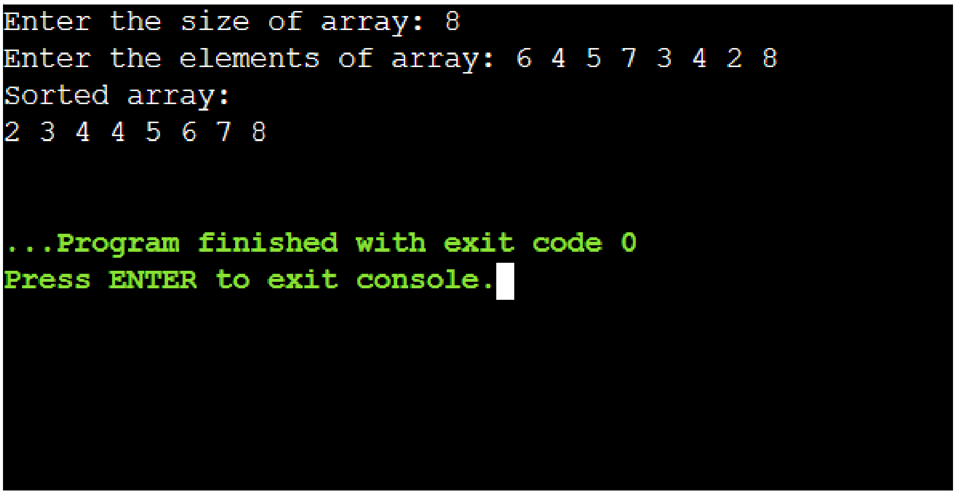
Merge Sort:

```
#include <stdio.h>

// Merge two subarrays L and M into arr
void merge(int arr[], int p, int q, int r) {

  // Create L ← A[p..q] and M ← A[q+1..r]
  int n1 = q - p + 1;
  int n2 = r - q;
  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];

  // Maintain current index of sub-arrays and main array
  int i, j, k;
  i = 0;
  j = 0;
  k = p;

  // Until we reach either end of either L or M, pick larger among
  // elements L and M and place them in the correct position at A[p..r]
  while (i < n1 && j < n2) {
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }
```

```
  // When we run out of elements in either L or M,
  // pick up the remaining elements and put in A[p..r]
  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}

// Divide the array into two subarrays, sort them and merge them
void mergeSort(int arr[], int l, int r) {
  if (l < r) {

    // m is the point where the array is divided into two subarrays
    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    // Merge the sorted subarrays
    merge(arr, l, m, r);
  }
}

// Print the array
void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++)
    printf("%d ", arr[i]);
  printf("\n");
}

// Driver program
int main() {
  int n;
  printf("Enter the size of array: ");
  scanf("%d",&n);

  int data[n];
  printf("Enter the elements of array: ");
  for(int i=0;i<n;i++)
    scanf("%d",&data[i]);
  mergeSort(data, 0, n - 1);

  printf("Sorted array: \n");
  printArray(data, n);
}
```

}

Output:



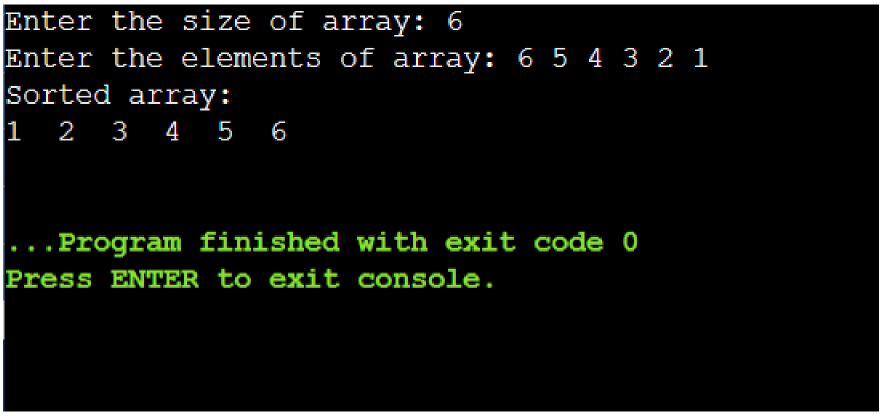
```
Enter the size of array: 8
Enter the elements of array: 6 4 5 7 3 4 2 8
Sorted array:
2 3 4 4 5 6 7 8


...Program finished with exit code 0
Press ENTER to exit console.
```

Shell Sort:

```
#include <stdio.h>
void shellSort(int array[], int n) {
  // Rearrange elements at each n/2, n/4, n/8, ... intervals
  for (int interval = n / 2; interval > 0; interval /= 2) {
    for (int i = interval; i < n; i += 1) {
      int temp = array[i];
      int j;
      for (j = i; j >= interval && array[j - interval] > temp; j -= interval) {
        array[j] = array[j - interval];
      }
      array[j] = temp;
    }
  }
}

// Print an array
```

```
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

// Driver code
int main() {
  int n;
  printf("Enter the size of array: ");
  scanf("%d",&n);
  int data[n];
  printf("Enter the elements of array: ");
  for(int i=0;i<n;i++)
    scanf("%d",&data[i]);
  shellSort(data, n);
  printf("Sorted array: \n");
  printArray(data, n);
}
```

Output:



```
Enter the size of array: 6
Enter the elements of array: 6 5 4 3 2 1
Sorted array:
1  2  3  4  5  6


...Program finished with exit code 0
Press ENTER to exit console.
```