Bubble Sort

```
#include <stdio.h>
// Function prototype
void bubble_sort(int arr[], int n);

int main() {
   int n;

   // Input the size of the array
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);

   int arr[n];

   // Input the elements of the array
   printf("Enter %d unsorted elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   // Perform Bubble Sort
   bubble_sort(arr, n);

   // Display the sorted array
   printf("Sorted array: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}

// Function to perform Bubble Sort
void bubble_sort(int arr[], int n) {
   for (int i = 0; i < n-1; i++) {
      for (int j = 0; j < n-i-1; j++) {
         if (arr[j] > arr[j+1]) {
            // Swap arr[j] and arr[j+1]
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
         }
      }
   }
}
```

Selection Sort

```
#include <stdio.h>
// Function prototype
void selection_sort(int arr[], int n);
int main() {
   int n;
   // Input the size of the array
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);

   int arr[n];

   // Input the elements of the array
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   // Perform Selection Sort
   selection_sort(arr, n);

   // Display the sorted array
   printf("Sorted array: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}

// Function to perform Selection Sort
void selection_sort(int arr[], int n) {
   for (int i = 0; i < n-1; i++) {
      int min_idx = i;
      for (int j = i+1; j < n; j++) {
         if (arr[j] < arr[min_idx]) {
            min_idx = j;
         }
      }
      // Swap arr[i] and arr[min_idx]
      int temp = arr[min_idx];
      arr[min_idx] = arr[i];
      arr[i] = temp;
   }
}
```

Insertion sort

```
#include <stdio.h>

// Function prototype
void insertion_sort(int arr[], int n);

int main() {
   int n;

   // Input the size of the array
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);

   int arr[n];

   // Input the elements of the array
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   // Perform Insertion Sort
   insertion_sort(arr, n);

   // Display the sorted array
   printf("Sorted array: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}

// Function to perform Insertion Sort
void insertion_sort(int arr[], int n) {
   for (int i = 1; i < n; i++) {
      int key = arr[i];
      int j = i - 1;
      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
```

Merge Sort

```
#include <stdio.h>

// Function prototypes
void merge_sort(int arr[], int l, int r);
void merge(int arr[], int l, int m, int r);

int main() {
   int n;

   // Input the size of the array
   printf("Enter the number of elements in the array: ");
   scanf("%d", &n);

   int arr[n];

   // Input the elements of the array
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   // Perform Merge Sort
   merge_sort(arr, 0, n - 1);

   // Display the sorted array
   printf("Sorted array: ");
   for (int i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");

   return 0;
}

// Function to perform Merge Sort
void merge_sort(int arr[], int l, int r) {
   if (l < r) {
      int m = l + (r - l) / 2;
      merge_sort(arr, l, m);
      merge_sort(arr, m + 1, r);
      merge(arr, l, m, r);
   }
}

// Function to merge two subarrays
```

```
void merge(int arr[], int l, int m, int r) {
   int n1 = m - l + 1;
   int n2 = r - m;
   int L[n1], R[n2];

   for (int i = 0; i < n1; i++) {
      L[i] = arr[l + i];
   }
   for (int j = 0; j < n2; j++) {
      R[j] = arr[m + 1 + j];
   }

   int i = 0, j = 0, k = l;
   while (i < n1 && j < n2) {
      if (L[i] <= R[j]) {
         arr[k] = L[i];
         i++;
      } else {
         arr[k] = R[j];
         j++;
      }
      k++;
   }

   while (i < n1) {
      arr[k] = L[i];
      i++;
      k++;
   }

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

Quick Sort

```
#include <stdio.h>

// Function prototypes
void quick_sort(int arr[], int low, int high);
int partition(int arr[], int low, int high);

int main() {
   int n;
```

```
    // Input the size of the array
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];

    // Input the elements of the array
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Perform Quick Sort
    quick_sort(arr, 0, n - 1);

    // Display the sorted array
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}

// Function to perform Quick Sort
void quick_sort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quick_sort(arr, low, pi - 1);
        quick_sort(arr, pi + 1, high);
    }
}

// Function to partition the array
int partition(int arr[], int low, int high) {
    int pivot = arr[low]; // Pivot is the first element
    int i = low;
    int j = high;

    while (i < j) {
        while (arr[i] <= pivot && i <= high - 1) {
            i++;
        }
        while (arr[j] > pivot && j >= low + 1) {
            j--;
        }
```

```
        if (i < j) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    // Swap pivot with element at index j
    int temp = arr[low];
    arr[low] = arr[j];
    arr[j] = temp;

    return j; // Return the partition point
}
```