SELECTION SORT

```
#include<stdio.h>
void selectionSort(int arr[], int n) {
    int i, j, minIndex, temp;
    for (i = 0; i < n-1; i++) {
        minIndex = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }
 // Swap the found minimum element with the first element
temp = arr[minIndex];
arr[minIndex] = arr[i];
arr[i] = temp;
 }}
int main() {
 int i, n, arr[100];
  printf("Enter the number of elements: ");
    scanf("%d", &n);
   printf("Enter the elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
   // Perform selection sort
    selectionSort(arr, n);
    printf("Sorted elements in ascending order:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

BUBBLE SORT

```
#include<stdio.h>
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        // Last i elements are already in place
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int main() {
    int i, n, arr[100];

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

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

    // Perform bubble sort
    bubbleSort(arr, n);

    printf("Sorted elements in ascending order:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

INSERTION SORT

```
#include <stdio.h>

void insertionSort(int arr[], int n) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }

        arr[j + 1] = key;
    }
}

int main() {
    int arr[100], n, i;

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

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

    insertionSort(arr, n);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

MERGE SORT

```
#include <stdio.h>

void merge(int arr[], int left[], int leftSize, int right[], int rightSize) {
    int i = 0, j = 0, k = 0;

    while (i < leftSize && j < rightSize) {
        if (left[i] <= right[j]) {
            arr[k] = left[i];
            i++;
        } else {
            arr[k] = right[j];
            j++;
        }
        k++;
    }

    while (i < leftSize) {
        arr[k] = left[i];
        i++;
        k++;
    }

    while (j < rightSize) {
        arr[k] = right[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int n) {
    if (n <= 1)
        return;

    int mid = n / 2;
    int left[mid];
    int right[n - mid];

    for (int i = 0; i < mid; i++) {
        left[i] = arr[i];
    }

    for (int i = mid; i < n; i++) {
        right[i - mid] = arr[i];
    }

    mergeSort(left, mid);
    mergeSort(right, n - mid);

    merge(arr, left, mid, right, n - mid);
}

int main() {
    int arr[100], n, i;

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

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

    mergeSort(arr, n);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

QUICK SORT

```
#include <stdio.h>

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int arr[100], n, i;

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

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

    quickSort(arr, 0, n - 1);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```
RADIX SORT
```
#include <stdio.h>

// Get the maximum value in the array
int getMax(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

// Counting sort based on a digit (exp)
void countingSort(int arr[], int n, int exp) {
    int output[n]; // Output array
    int count[10] = {0}; // Initialize count array with all zeros

    // Store count of occurrences in count[]
    for (int i = 0; i < n; i++) {
        count[(arr[i] / exp) % 10]++;
    }

    // Change count[i] so that count[i] now contains
    // actual position of this digit in output[]
    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }

    // Build the output array
    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    // Copy the output array to arr[]
    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
}

// Radix Sort
void radixSort(int arr[], int n) {
    // Find the maximum number to determine the number of digits
    int max = getMax(arr, n);

    // Perform counting sort for every digit
    for (int exp = 1; max / exp > 0; exp *= 10) {
        countingSort(arr, n, exp);
    }
}

int main() {
    int arr[100], n, i;

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

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

    radixSort(arr, n);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

ALPHABETIC DATA:

SELECTION SORT

```
#include <stdio.h>
#include <string.h>

void selectionSort(char arr[][100], int n) {
    int i, j, minIndex;
    char temp[100];

    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (strcmp(arr[j], arr[minIndex]) < 0)
                minIndex = j;
        }

        if (minIndex != i) {
            strcpy(temp, arr[i]);
            strcpy(arr[i], arr[minIndex]);
            strcpy(arr[minIndex], temp);
        }
    }
}

int main() {
    char arr[100][100];
    int n, i;

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

    printf("Enter elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%s", arr[i]);
    }

    selectionSort(arr, n);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%s ", arr[i]);
    }

    return 0;
}
```

BUBBLE SORT

```
#include <stdio.h>
#include <string.h>

void bubbleSort(char arr[][100], int n) {
    int i, j;
    char temp[100];

    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (strcmp(arr[j], arr[j+1]) > 0) {
                strcpy(temp, arr[j]);
                strcpy(arr[j], arr[j+1]);
                strcpy(arr[j+1], temp);
            }
        }
    }
}

int main() {
    char arr[100][100];
    int n, i;

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

    printf("Enter elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%s", arr[i]);
    }

    bubbleSort(arr, n);

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%s ", arr[i]);
    }

    return 0;
}
```