Bubble Sort

```
#include <iostream>
using namespace std;

void bubbleSort(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], arr[j + 1]);
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Insertion Sort

```
#include <iostream>
using namespace std;

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i], j = i - 1;
        while (j >= 0 && arr[j] > key)
            arr[j + 1] = arr[j--];
        arr[j + 1] = key;
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Selection Sort

```
#include <iostream>
using namespace std;

void selectionSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIdx = i;
        for (int j = i + 1; j < n; j++)
            if (arr[j] < arr[minIdx])
                minIdx = j;
        swap(arr[i], arr[minIdx]);
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {29, 10, 14, 37, 13};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Quick Sort

```
#include <iostream>
using namespace std;

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

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
```

```
    int arr[] = {10, 7, 8, 9, 1, 5};
    int n = sizeof(arr)/sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    printArray(arr, n);
    return 0;
}
```

Merge Sort

```
#include <iostream>
using namespace std;

void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1, 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) arr[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
    while (i < n1) arr[k++] = L[i++];
    while (j < n2) arr[k++] = R[j++];
}

void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr)/sizeof(arr[0]);
    mergeSort(arr, 0, n - 1);
    printArray(arr, n);
    return 0;
}
```

Heap Sort

```
#include <iostream>
using namespace std;

void heapify(int arr[], int n, int i) {
    int largest = i, l = 2*i+1, r = 2*i+2;
    if (l < n && arr[l] > arr[largest]) largest = l;
    if (r < n && arr[r] > arr[largest]) largest = r;
```

```
    if (largest != i) {
        swap(arr[i], arr[largest]);
        heapify(arr, n, largest);
    }
}

void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);
    for (int i = n - 1; i > 0; i--) {
        swap(arr[0], arr[i]);
        heapify(arr, i, 0);
    }
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {4, 10, 3, 5, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    heapSort(arr, n);
    printArray(arr, n);
    return 0;
}
```

Linear Search

```
#include <iostream>
using namespace std;

int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++)
        if (arr[i] == key)
            return i;
    return -1;
}

int main() {
    int arr[] = {5, 8, 12, 20, 34};
    int n = sizeof(arr)/sizeof(arr[0]), key;
    cin >> key;
    int result = linearSearch(arr, n, key);
    if (result != -1) cout << "Found at index " << result << endl;
    else cout << "Not found." << endl;
    return 0;
}
```

Binary Search

```
#include <iostream>
using namespace std;
```

```
int binarySearch(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = low + (high - low)/2;
        if (arr[mid] == key) return mid;
        else if (arr[mid] < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main() {
    int arr[] = {2, 4, 7, 10, 15, 20, 25};
    int n = sizeof(arr)/sizeof(arr[0]), key;
    cin >> key;
    int result = binarySearch(arr, n, key);
    if (result != -1) cout << "Found at index " << result << endl;
    else cout << "Not found." << endl;
    return 0;
}
```