```cpp
#include <iostream>
#include <vector>
#include <algorithm>

class Search {
public:
    static int linearSearch(const std::vector<int>& arr, int key) {
        for (int i = 0; i < arr.size(); ++i) {
            if (arr[i] == key) {
                return i; // اگر عنصر پیدا شد، اندیس آن را برمی‌گردانیم
            }
        }
        return -1; // اگر عنصر پیدا نشد، -1 را برمی‌گردانیم
    }

    static int binarySearch(const std::vector<int>& arr, int key) {
        int left = 0;
        int right = arr.size() - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == key) {
                return mid; // اگر عنصر پیدا شد، اندیس آن را برمی‌گردانیم
            }
            if (arr[mid] < key) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1; // اگر عنصر پیدا نشد، -1 را برمی‌گردانیم
    }

    static int mergeSearch(const std::vector<int>& arr, int key) {
        std::vector<int> sortedArr = arr; // ایجاد یک کپی از آرایه
        std::sort(sortedArr.begin(), sortedArr.end()); // مرتب‌سازی کپی شده
        int result = binarySearch(sortedArr, key); // سرچ دودویی در آرایه مرتب‌شده
```

```
        if (result != -1) {
            // یافتن اندیس متناظر در آرایه اصلی
            for (int i = 0; i < arr.size(); ++i) {
                if (arr[i] == sortedArr[result]) {
                    return i;
                }
            }
        }
        return -1; // اگر عنصر پیدا نشد، -1 را برمی‌گردانیم
    }

    static int selectionSearch(const std::vector<int>& arr, int key) {
        for (int i = 0; i < arr.size(); ++i) {
            int minIndex = i;
            for (int j = i + 1; j < arr.size(); ++j) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }
            if (arr[minIndex] == key) {
                return minIndex; // اگر عنصر پیدا شد، اندیس آن را برمی‌گردانیم
            }
        }
        return -1; // اگر عنصر پیدا نشد، -1 را برمی‌گردانیم
    }
};

class Sort {
public:
    static void bubbleSort(std::vector<int>& arr) {
        int n = arr.size();
        for (int i = 0; i < n - 1; ++i) {
            for (int j = 0; j < n - i - 1; ++j) {
                if (arr[j] > arr[j + 1]) {
                    std::swap(arr[j], arr[j + 1]);
                }
            }
        }
```

```
    }

    static void insertionSort(std::vector<int>& arr) {
        int n = arr.size();
        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;
        }
    }

    static void mergeSort(std::vector<int>& arr) {
        if (arr.size() > 1) {
            int mid = arr.size() / 2;
            std::vector<int> left(arr.begin(), arr.begin() + mid);
            std::vector<int> right(arr.begin() + mid, arr.end());
            mergeSort(left);
            mergeSort(right);
            merge(arr, left, right);
        }
    }

    static void merge(std::vector<int>& arr, const std::vector<int>& left, const std::vector<int>&
right) {
        int nL = left.size();
        int nR = right.size();
        int i = 0, j = 0, k = 0;
        while (i < nL && j < nR) {
            if (left[i] <= right[j]) {
                arr[k] = left[i];
                i++;
            } else {
                arr[k] = right[j];
                j++;
```

```
            }
            k++;
        }

while (i < nL) {
            arr[k] = left[i];
            i++;
            k++;
        }
        while (j < nR) {
            arr[k] = right[j];
            j++;
            k++;
        }
    }

    static void selectionSort(std::vector<int>& arr) {
        int n = arr.size();
        for (int i = 0; i < n - 1; ++i) {
            int minIndex = i;
            for (int j = i + 1; j < n; ++j) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }
            std::swap(arr[i], arr[minIndex]);
        }
    }
};

int main() {
    std::vector<int> numbers = {64, 34, 25, 12, 22, 11, 90};
    int key = 22;

    // سرچ خطی
    int resultLinear = Search::linearSearch(numbers, key);
    if (resultLinear != -1) {
        std::cout << "عنصر با مقدار " << key << " در اندیس " << resultLinear << " پیدا شد." << std::endl;
```

```
    } else {
        std::cout << "عنصر با مقدار " << key << " پیدا نشد." << std::endl;
    }

    // سرچ دودویی
    int resultBinary = Search::binarySearch(numbers, key);
    if (resultBinary != -1) {
        std::cout << "عنصر با مقدار " << key << " در اندیس " << resultBinary << " پیدا شد." << std::endl;
    } else {
        std::cout << "عنصر با مقدار " << key << " پیدا نشد." << std::endl;
    }

    // سرچ با مرتب‌سازی و سپس سرچ دودویی
    int resultMerge = Search::mergeSearch(numbers, key);
    if (resultMerge != -1) {
        std::cout << "عنصر با مقدار " << key << " در اندیس " << resultMerge << " پیدا شد." << std::endl;
    } else {
        std::cout << "عنصر با مقدار " << key << " پیدا نشد." << std::endl;
    }

    // سرچ با الگوریتم سورت شده
    int resultSelection = Search::selectionSearch(numbers, key);
    if (resultSelection != -1) {
        std::cout << "عنصر با مقدار " << key << " در اندیس " << resultSelection << " پیدا شد." <<
std::endl;
    } else {
        std::cout << "عنصر با مقدار " << key << " پیدا نشد." << std::endl;
    }

    // مرتب‌سازی با Bubble Sort
    Sort::bubbleSort(numbers);

    // مرتب‌سازی با Insertion Sort
    Sort::insertionSort(numbers);

    // مرتب‌سازی با Merge Sort
    Sort::mergeSort(numbers);
```

```
    // مرتب‌سازی با Selection Sort
    Sort::selectionSort(numbers);

    return 0;
}
```