searchSort.h

```
class SearchSort
{
private:
public:
    int linearSearch(const int arr[], int size, int value);
    int binarySearch(const int array[], int size, int value);
    void bubbleSort(int array[], int size);
    void selectionSort(int array[], int size, int &swaps);
    void swap(int &a, int &b);
};
```

searchSort.cpp

```
#include "searchSort.h"
#include <iostream>

using namespace std;

//**************************************************************
***
// The linearSearch function performs a linear search on an
*
// integer array. The array arr, which has a maximum of size
*
// elements, is searched for the number stored in value. If the
*
// number is found, its array subscript is returned. Otherwise,
*
// -1 is returned indicating the value was not in the array.
*
//**************************************************************
***
```

```cpp
int SearchSort::linearSearch(const int arr[], int size, int
value)
{
    int index = 0;      // Used as a subscript to search array
    int position = -1;  // To record position of search value
    bool found = false; // Flag to indicate if the value was
found

    while (index < size && !found)
    {
        if (arr[index] == value) // If the value is found
        {
            found = true;      // Set the flag
            position = index; // Record the value's subscript
        }
        index++; // Go to the next element
    }
    return position; // Return the position, or -1
}

//*****************************************************************
*
// The binarySearch function performs a binary search on an
*
// integer array. array, which has a maximum of size elements,
*
// is searched for the number stored in value. If the number is
*
// found, its array subscript is returned. Otherwise, -1 is
*
// returned indicating the value was not in the array.
*
```

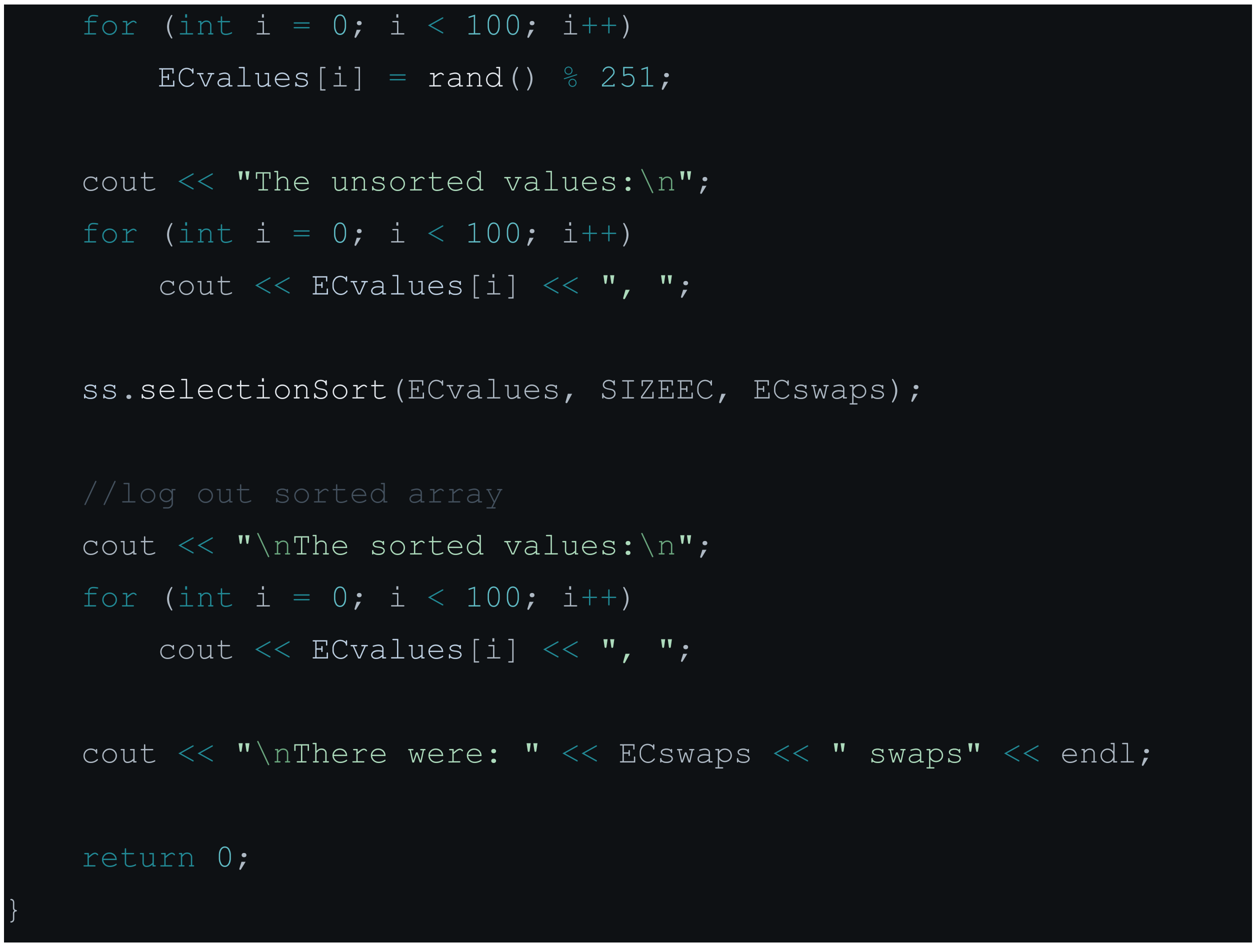
```
//**************************************************************
*

int SearchSort::binarySearch(const int array[], int size, int
value)
{
    int first = 0,       // First array element
        last = size - 1, // Last array element
        middle,          // Mid point of search
        position = -1;   // Position of search value
    bool found = false;  // Flag

    while (!found && first <= last)
    {
        middle = (first + last) / 2; // Calculate mid point
        if (array[middle] == value)  // If value is found at mid
        {
            found = true;
            position = middle;
        }
        else if (array[middle] > value) // If value is in lower
half
            last = middle - 1;
        else
            first = middle + 1; // If value is in upper half
    }
    return position;
}

//**************************************************************
***
// The bubbleSort function sorts an int array in ascending
order. *
```

```
//*****************************************************************
***
void SearchSort::bubbleSort(int array[], int size)
{
    int maxElement;
    int index;

    for (maxElement = size - 1; maxElement > 0; maxElement--)
    {
        for (index = 0; index < maxElement; index++)
        {
            if (array[index] > array[index + 1])
            {
                swap(array[index], array[index + 1]);
            }
        }
    }
}

//*****************************************************************
******
// The selectionSort function sorts an int array in ascending
order. *
//*****************************************************************
******
void SearchSort::selectionSort(int array[], int size, int
&swaps)
{
    swaps = 0;
    int minIndex, minValue;

    for (int start = 0; start < (size - 1); start++)
    {
        minIndex = start;
        minValue = array[start];
        for (int index = start + 1; index < size; index++)
        {
            if (array[index] < minValue)
            {
                swaps++;
                minValue = array[index];
                minIndex = index;
            }
        }
        swap(array[minIndex], array[start]);
    }
}

//*****************************************************
// The swap function swaps a and b in memory.        *
//*****************************************************
void SearchSort::swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

Main.cpp

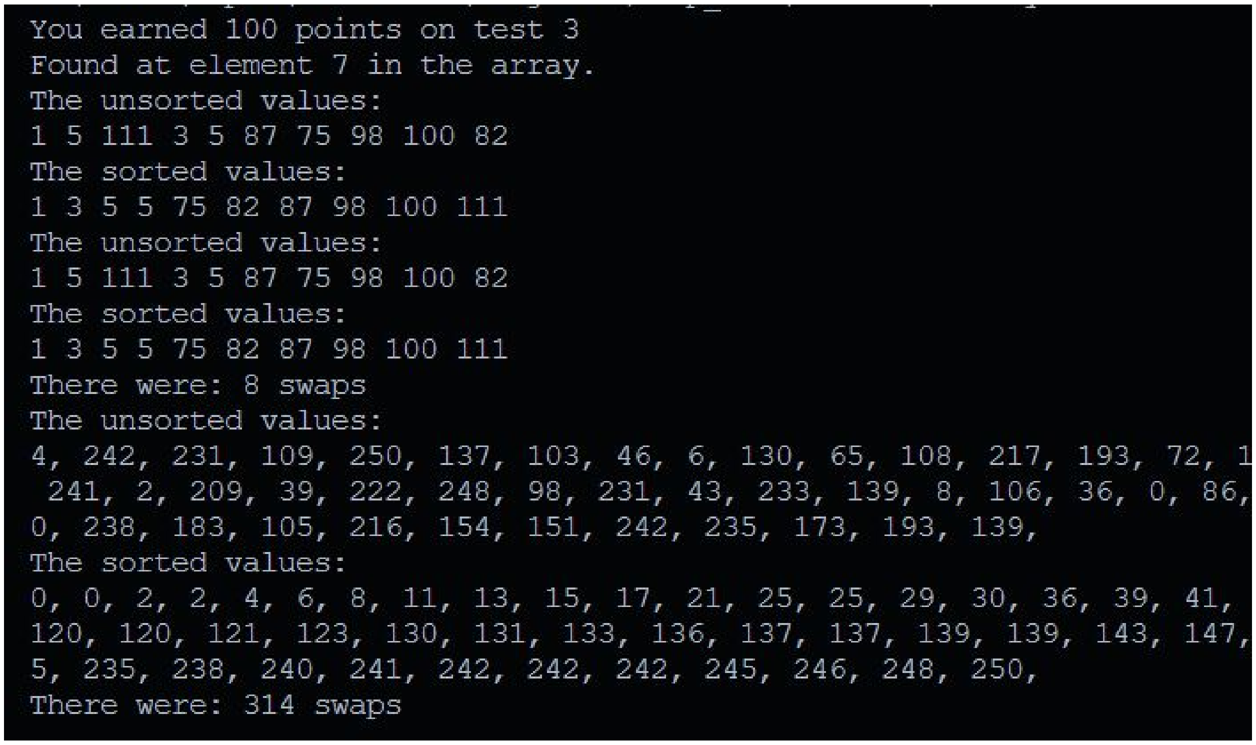
```
#include "searchSort.h"
#include <iostream>
#include <ctime>

using namespace std;
```

```cpp
int main()
{
    SearchSort ss;

    //linear search
    const int SIZE = 10;
    int array[SIZE] = {1, 5, 111, 3, 5, 87, 75, 98, 100, 82};
    int results;

    results = ss.linearSearch(array, SIZE, 111);

    if (results == -1)
        cout << "You did not earn 100 points on any test\n";
    else
    {
        cout << "You earned 100 points on test ";
        cout << (results + 1) << endl;
    }

    results = ss.binarySearch(array, SIZE, 98);

    if (results == -1)
        cout << "That number does not exist in the array.\n";
    else
        cout << "Found at element " << results << " in the
array." << endl;

    //bubble sort
    int bubbleSortArray[SIZE] = {1, 5, 111, 3, 5, 87, 75, 98,
100, 82};
    cout << "The unsorted values:\n";
    for (auto element : bubbleSortArray)
        cout << element << " ";
    cout << endl;

    ss.bubbleSort(bubbleSortArray, SIZE);

    cout << "The sorted values:\n";
    for (auto element : bubbleSortArray)
        cout << element << " ";
    cout << endl;

    //selection sort
    int selectionSortArray[SIZE] = {1, 5, 111, 3, 5, 87, 75, 98,
100, 82};
    cout << "The unsorted values:\n";
    for (auto element : selectionSortArray)
        cout << element << " ";
    cout << endl;

    int s;
    ss.selectionSort(selectionSortArray, SIZE, s);

    cout << "The sorted values:\n";
    for (auto element : selectionSortArray)
        cout << element << " ";
    cout << endl;
    cout << "There were: " << s << " swaps" << endl;

    //Extra credit
    const int SIZEEC = 100;
    int ECswaps;
    int ECvalues[SIZEEC];
    srand(time(0));

    //create random generated array
```



```cpp
    for (int i = 0; i < 100; i++)
        ECvalues[i] = rand() % 251;

    cout << "The unsorted values:\n";
    for (int i = 0; i < 100; i++)
        cout << ECvalues[i] << ", ";

    ss.selectionSort(ECvalues, SIZEEC, ECswaps);

    //log out sorted array
    cout << "\nThe sorted values:\n";
    for (int i = 0; i < 100; i++)
        cout << ECvalues[i] << ", ";

    cout << "\nThere were: " << ECswaps << " swaps" << endl;

    return 0;
}
```

Output cut down to fit in document. It does display 100 numbers



```
You earned 100 points on test 3
Found at element 7 in the array.
The unsorted values:
1 5 111 3 5 87 75 98 100 82
The sorted values:
1 3 5 5 75 82 87 98 100 111
The unsorted values:
1 5 111 3 5 87 75 98 100 82
The sorted values:
1 3 5 5 75 82 87 98 100 111
There were: 8 swaps
The unsorted values:
4, 242, 231, 109, 250, 137, 103, 46, 6, 130, 65, 108, 217, 193, 72, 1
 241, 2, 209, 39, 222, 248, 98, 231, 43, 233, 139, 8, 106, 36, 0, 86,
0, 238, 183, 105, 216, 154, 151, 242, 235, 173, 193, 139,
The sorted values:
0, 0, 2, 2, 4, 6, 8, 11, 13, 15, 17, 21, 25, 25, 29, 30, 36, 39, 41,
120, 120, 121, 123, 130, 131, 133, 136, 137, 137, 139, 139, 143, 147,
5, 235, 238, 240, 241, 242, 242, 242, 245, 246, 248, 250,
There were: 314 swaps
```