```
Script started on 2025-07-21 14:06:19-05:00 [TERM="xterm" TTY="/dev/pts/0" COLUMNS=
kn55307@ares:~/Portfolio 3/sorting lab$ pwd

/home/students/kn55307/Portfolio 3/sorting lab
kn55307@ares:~/Portfolio 3/sorting lab$ cat sorts.info

Noah Kang
CSC121-001
Sorts (Order is the key...) Lab
Level 6:
    Level 4: Base Level
    Level 0.5: Allows users to specify the bounds of the range of the
        random values that the vector is being filled with
    Level 1.5: Added a menu to allow the user to choose which sorting
        algorithm to test

This program allows users to test selection and insertion sort methods
on a vector filled with randomly generated values.
kn55307@ares:~/Portfolio 3/sorting lab$ show-code sorts_driver.cpp



sorts_driver.cpp:
```

```cpp
#include "sorts.h"
#include "sorts.cpp"
#include <iostream>
#include <string>
#include <vector>
using namespace std;

vector<int> generateRandomVector(int size, int lowerBound,
    int upperBound) {
    vector<int> v(size);
    for (int i = 0; i < size; ++i) {
        v[i] = rand() % (upperBound - lowerBound + 1) + lowerBound;
    }
    return v;
}

int main(){
    cout << "Welcome to the Sorting Lab!" << endl;
    bool running = true;
    while (running){
        cout << "Would you like to sort a vector? (y/n): ";
        string input;
        getline(cin, input);
        if (input[0] == 'y') {
            // Call sorting function
            cout << "Enter the size of the vector: ";
            int vectorSize;
            cin >> vectorSize;
            cin.ignore();
            cout << "Enter the upper bound: ";
            int upperBound;
            cin >> upperBound;
            cin.ignore();
            cout << "Enter the lower bound: ";
            int lowerBound;
            cin >> lowerBound;
            cin.ignore();

            cout << "Choose a sorting algorithm (1 for Selection Sort,"
                << " 2 for Insertion Sort): ";
            int choice;
            cin >> choice;
            cin.ignore();

            cout << "Would you like to sort within a specific range?"
                << " (y/n): ";
            string rangeInput;
            bool isRanged = true;
            getline(cin, rangeInput);
            int rangeStart, rangeEnd;
            if (rangeInput[0] == 'y') {
                cout << "Enter the sorting range of the vector: " << endl;
                cout << "Start index: ";
                cin >> rangeStart;
                cin.ignore();
                cout << "End index: ";
                cin >> rangeEnd;
                cin.ignore();
            } else {
                isRanged = false;
            }


            vector<int> v = generateRandomVector(vectorSize, lowerBound,
                upperBound);
            cout << "Generated vector: ";
            for (const auto& num : v) {
                cout << num << " ";
            }
            cout << endl;

            if (choice == 1) {
                cout << "You chose Selection Sort." << endl;
                if (isRanged){
                    selectionSort(v, rangeStart, rangeEnd);
                } else {
                    selectionSort(v);
                }
                cout << "Sorted vector: ";
                for (const auto& num : v) {
                    cout << num << " ";
                }
                cout << endl;
            } else if (choice == 2) {
                cout << "You chose Insertion Sort." << endl;
                if (isRanged){
                    insertionSort(v, rangeStart, rangeEnd);
                } else {
                    insertionSort(v);
                }
                cout << "Sorted vector: ";
                for (const auto& num : v) {
                    cout << num << " ";
                }
                cout << endl;
            } else {
                cout << "Invalid choice. Exiting." << endl;
                break;
            }


        } else {
            cout << "Have a great day!" << endl;
            running = false;
        }
    }


    return 0;
}
kn55307@ares:~/Portfolio 3/sorting lab$ show-code sorts.cpp



sorts.cpp:


#include "sorts.h"

// Selection Sort (range)
void selectionSort(vector<int>& v, size_t begin, size_t end) {
    for (size_t i = begin; i < end; ++i) {
        size_t minIndex = i;
        for (size_t j = i + 1; j < end; ++j) {
            if (v[j] < v[minIndex]){
                minIndex = j;
            }
        }
        if (minIndex != i){
            std::swap(v[i], v[minIndex]);
        }
    }
}

// Selection Sort (whole vector)
inline void selectionSort(vector<int>& v) {
    selectionSort(v, 0, v.size());
}

// Insertion Sort (range)
void insertionSort(vector<int>& v, size_t begin, size_t end) {
    for (size_t i = begin + 1; i < end; ++i) {
        int key = v[i];
        size_t j = i;
        while (j > begin && v[j - 1] > key) {
            v[j] = v[j - 1];
            --j;
        }
        v[j] = key;
    }
}

// Insertion Sort (whole vector)
inline void insertionSort(vector<int>& v) {
    insertionSort(v, 0, v.size());
}
kn55307@ares:~/Portfolio 3/sorting lab$ CPP sorts

sorts.cpp...
No main found...simply generating .o files...


kn55307@ares:~/Portfolio 3/sorting lab$ CPP sorts_driver

sorts_driver.cpp***


kn55307@ares:~/Portfolio 3/sorting lab$ ./sorts_driver.out

Welcome to the Sorting Lab!
Would you like to sort a vector? (y/n): y
Enter the size of the vector: 10
Enter the upper bound: 20
Enter the lower bound: 0
Choose a sorting algorithm (1 for Selection Sort, 2 for Insertion Sort): 1
Would you like to sort within a specific range? (y/n): n
Generated vector: 1 4 9 19 8 10 10 9 15 10
You chose Selection Sort.
Sorted vector: 1 4 8 9 9 10 10 10 15 19
Would you like to sort a vector? (y/n): y
Enter the size of the vector: 12
Enter the upper bound: 50
Enter the lower bound: 0
Choose a sorting algorithm (1 for Selection Sort, 2 for Insertion Sort): 1
Would you like to sort within a specific range? (y/n): y
Enter the sorting range of the vector:
Start index: 0
End index: 4
Generated vector: 5 31 20 19 29 22 45 30 40 31 35 8
You chose Selection Sort.
Sorted vector: 5 19 20 31 29 22 45 30 40 31 35 8
```

```
Would you like to sort a vector? (y/n): y
Enter the size of the vector: 10
Enter the upper bound: 60
Enter the lower bound: 0
Choose a sorting algorithm (1 for Selection Sort, 2 for Insertion Sort): 2
Would you like to sort within a specific range? (y/n): n
Generated vector: 38 8 30 3 15 48 60 58 24 30
You chose Insertion Sort.
Sorted vector: 3 8 15 24 30 30 38 48 58 60
Would you like to sort a vector? (y/n): y
Enter the size of the vector: 10
Enter the upper bound: 50
Enter the lower bound: 0
Choose a sorting algorithm (1 for Selection Sort, 2 for Insertion Sort): 2
Would you like to sort within a specific range? (y/n): y
Enter the sorting range of the vector:
Start index: 0
End index: 5
Generated vector: 27 0 23 33 37 10 4 31 3 9
You chose Insertion Sort.
Sorted vector: 0 23 27 33 37 10 4 31 3 9
Would you like to sort a vector? (y/n): n
Have a great day!
kn55307@ares:~/Portfolio 3/sorting lab$ cat sorts.tpq

1. Explain (briefly) the differing philosophies of selection and insertion
    sort. How can they both produce a sorted list?

    Selection sort finds the minimum value in subsets of decreasing size
    while insertion sort mainly "inserts" the minimum value at each
    increasing index of the list.

2. What type of values are your functions sorting? Does it really matter
    what type it is in terms of the algorithms? Does it make very much of
    a difference to your implementation of the algorithms? How easily
    could you change the type of information you were sorting? (Is there a
    special type of data that might be more difficult to adjust to?)

    My functions sort integer values, but it the data type doesn't really
    matter in terms of the algorithms, as long as they are some form of
    a numerical type.

3. Does it matter what range of values you produce to randomly fill the
    vector? Will this affect the sorting in any way?

    The range of the values doesn't necessarily affect the sorting in
    any way.

4. Do your functions sort into ascending or descending order? How would
    you go about changing this order? As long as it is 'in order', does it
    truly matter which direction they are going?

    My functions sort in ascending order, but this could be changed by
    finding the maximum values and swapping them instead of minimum
    values.

5. How can you do the verification that the data is sorted? (Note that
    your program is supposed to do this automatically – not ask the user
    if the data is sorted after printing it!)

    You could sort the data using both methods and compare if the sorted
    lists of the two methods are equal.
kn55307@ares:~/Portfolio 3/sorting lab$ exit

exit

Script done on 2025-07-21 14:08:43-05:00 [COMMAND_EXIT_CODE="0"]
```