CSCI 230 PA 3 Submission

Due Date: 03/15/2022 Late (date and time):_________________

Name(s): Nero Li

Exercise 1 (with extra credit) -- need to submit source code and I/O

-- check if completely done ✓; otherwise, discuss issues below

Source code below:
exercise_1.cpp:

```
/*  Program: PA_3_exercise_1
    Author: Nero Li
    Class: CSCI 230
    Date: 03/15/2022
    Description:
        Implement Insertion Sort and collect data regarding number of
        key comparisons and data moves for a small list of 10 integer
        values. You must try a sorted list, a descending list, and a
        random list. In addition, collect data for a random list of 100
        values.

    I certify that the code below is my own work.

    Exception(s): N/A

*/

#include <iostream>
#include <vector>
#include <ctime>
#include <chrono>

using namespace std;

vector<int> generateVector(int n, int choice)
{
    vector<int> vec;

    for (int i = 0; i < n; ++i)
    {
        switch (choice)
```

```cpp
        {
            case 0:
                vec.push_back(i);
                break;
            case 1:
                vec.push_back(n - 1 - i);
                break;
            default:
                vec.push_back(rand() % n);
                break;
        }
    }

    return vec;
}

void printVector(vector<int> vec)
{
    for (int i : vec)
        cout << i << " ";
    cout << endl;
}

void insertionSort(vector<int> &vec, long long &comparisons, long long
&dataMoves)
{


    for (int i = 1; i < vec.size(); ++i)
    {
        int cur = vec[i];
        for (int j = 0; j < i; ++j)
        {
            ++comparisons;
            if (vec[i] < vec[j])
            {
                vector<int>::iterator a = vec.begin() + i;
                vector<int>::iterator b = vec.begin() + j;
                vec.erase(a);
                vec.insert(b, cur);
                ++dataMoves;
                break;
            }
        }
    }
```

```cpp
}

void test(string str, int n, int choice, bool checkTime, bool printVec)
{
    vector<int> vec = generateVector(n, choice);
    long long comparisons = 0;
    long long dataMoves = 0;

    cout << str << ":\n";

    if (printVec)
        printVector(vec);

    auto start = chrono::high_resolution_clock::now();
    insertionSort(vec, comparisons, dataMoves);
    auto end = chrono::high_resolution_clock::now();

    if (printVec)
        printVector(vec);

    cout << "Comparisons:\t" << comparisons << endl;
    cout << "Data moves:\t" << dataMoves << endl;
    if (checkTime)
        cout << "Time used:\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << endl;
}

int main()
{
    srand(time(NULL));

    test("Sorted list", 10, 0, false, true);
    test("Descending list", 10, 1, false, true);
    test("Random list", 10, 2, false, true);
    test("Large random list", 100, 2, false, false);

    test("1000 random list", 1000, 2, true, false);
    test("10000 random list", 10000, 2, true, false);
    test("100000 random list", 100000, 2, true, false);
    cout << "Author: Nero Li\n";

    return 0;
}
```

Input/output below:

```
Sorted list:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
Comparisons:    45
Data moves:     0

Descending list:
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
Comparisons:    9
Data moves:     9

Random list:
4 3 2 0 9 4 6 9 5 4
0 2 3 4 4 4 5 6 9 9
Comparisons:    37
Data moves:     7

Large random list:
Comparisons:    2426
Data moves:     93

1000 random list:
Comparisons:    244739
Data moves:     989
Time used:      2.0008 ms

10000 random list:
Comparisons:    25165195
Data moves:     9987
Time used:      133.12 ms

100000 random list:
Comparisons:    2500156973
Data moves:     99978
Time used:      13923.2 ms

Author: Nero Li
```

Exercise 2 (with extra credit) -- need to submit source code and I/O

-- check if completely done ✓ ; otherwise, discuss issues below

Source code below:
exercise_2.cpp:

```
/*  Program: PA_2_exercise_2
    Author: Nero Li
    Class: CSCI 230
    Date: 03/15/2022
    Description:
        Implement/use Merge Sort from the book and test it on a small
        list of 10 random integer values first. Collect data regarding
        number of key comparisons and data moves for a small list of 10
        integer values. You must try a sorted list, a descending list,
        and a random list. In addition, collect data for a random list
        of 100 values.

    I certify that the code below is my own work.

    Exception(s): N/A

*/

#include <iostream>
#include <vector>
#include <ctime>
#include <chrono>

using namespace std;

vector<int> generateVector(int n, int choice)
{
    vector<int> vec;

    for (int i = 0; i < n; ++i)
    {
        switch (choice)
        {
            case 0:
                vec.push_back(i);
                break;
            case 1:
                vec.push_back(n - 1 - i);
                break;
```

```cpp
            default:
                vec.push_back(rand() % n);
                break;
        }
    }

    return vec;
}

void printVector(vector<int> vec)
{
    for (int i : vec)
        cout << i << " ";
    cout << endl;
}

void merge(vector<int> &vec, long long &comparisons, long long &dataMoves,
vector<int> a, vector<int> b)
{
    int i = 0, j = 0;

    while (i < a.size() && j < b.size())
    {
        ++comparisons;
        if (a[i] < b[j])
            vec.push_back(a[i++]);
        else
        {
            ++dataMoves;
            vec.push_back(b[j++]);
        }
    }

    while (i < a.size() || j < b.size())
    {
        if (i < a.size())
            vec.push_back(a[i++]);
        else
            vec.push_back(b[j++]);
    }
}

void mergeSort(vector<int> &vec, long long &comparisons, long long
&dataMoves)
{
    if (vec.size() <= 1)
        return;

    vector<int> a;
    vector<int> b;

    for (int i = 0; i < vec.size(); ++i)
    {
        if (i < vec.size() / 2)
            a.push_back(vec[i]);
        else
            b.push_back(vec[i]);
    }

    mergeSort(a, comparisons, dataMoves);
    mergeSort(b, comparisons, dataMoves);
    vec.clear();
    merge(vec, comparisons, dataMoves, a, b);
}

void test(string str, int n, int choice, bool checkTime, bool printVec)
{
    vector<int> vec = generateVector(n, choice);
    long long comparisons = 0;
    long long dataMoves = 0;

    cout << str << ":\n";

    if (printVec)
        printVector(vec);

    auto start = chrono::high_resolution_clock::now();
    mergeSort(vec, comparisons, dataMoves);
    auto end = chrono::high_resolution_clock::now();

    if (printVec)
        printVector(vec);

    cout << "Comparisons:\t" << comparisons << endl;
    cout << "Data moves:\t" << dataMoves << endl;
    if (checkTime)
        cout << "Time used:\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;
```

```
    cout << endl;
}

int main()
{
    srand(time(NULL));

    test("Sorted list", 10, 0, false, true);
    test("Descending list", 10, 1, false, true);
    test("Random list", 10, 2, false, true);
    test("Large random list", 100, 2, false, false);

    test("1000 random list", 1000, 2, true, false);
    test("10000 random list", 10000, 2, true, false);
    test("100000 random list", 100000, 2, true, false);

    cout << "Author: Nero Li\n";

    return 0;
}
```

Input/output below:

```
Sorted list:
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
Comparisons:    15
Data moves:     0

Descending list:
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
Comparisons:    19
Data moves:     19

Random list:
3 6 2 6 8 0 0 9 3 6
0 0 2 3 3 6 6 6 8 9
Comparisons:    22
Data moves:     9

Large random list:
Comparisons:    539
Data moves:     270

1000 random list:
Comparisons:    8679
Data moves:     4355
Time used:      4.0021 ms
```

```
10000 random list:
Comparisons:    120319
Data moves:     61023
Time used:      42.0393 ms

100000 random list:
Comparisons:    1536198
Data moves:     776607
Time used:      1087.35 ms

Author: Nero Li
```

Answer for Question 1:

For my insertion sort, I find the first number that is bigger than my current number and then do the insert function, so if there is no number bigger than my current, it won't to the data moves and go to see the next number. Because of that, for a sorted list, it will do the most comparisons but no data moves, which was shown in my output; for a descending list, it will show fewer comparisons since it will be easier to find the bigger number and then do the insert function, which was shown in my output. Hence, the counts in my code seem reasonable.

Answer for Question 2:

For my merge sort, I did the recursion to help me divide the list of numbers into several groups. Then, to compare each group, we have a sub-vector a that is initially at the front and a sub-vector b that indicates the rest of the number. It means for each time, we compare two elements from both sub-vector and insert them into a new vector. If we get a comparison result that grabbed a number from vector b, that means we changed the initial place for one number, increase our data moves count. Based on this idea, for a sorted list, it will do fewer comparisons since for new vector, we keep grabbing numbers from vector a, which also means no data moves; for a descending list, it will show more comparisons and the same value of data moves counts since we did more insertion after each comparison, and we grabbed more data from our vector b instead of vector a. Hence, the counts in my code seem reasonable.