CSCI 230 PA 4 Submission

Due Date: 03/22/2022 Late (date and time):__________________

Name(s): Nero Li

Exercise 1 -- need to submit source code and I/O

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

Source code below:

exercise_1.cpp:

```
/*  Program: PA_4_exercise_1
    Author: Nero Li
    Class: CSCI 230
    Date: 03/22/2022
    Description:
        Implement/use Quick 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>

using namespace std;


void quickSortStep(std::vector<int>& S, int a, int b, long long
&comparisons, long long &dataMoves)
{
    if (a >= b) return;                                             //
0 or 1 left? done
    ++dataMoves;
    int pivot = S[b];                                                       //
select last as pivot
    int l = a;                                                              //
left edge
    int r = b - 1;                                                          //
right edge
    while (l <= r)
    {
        while (l <= r && pivot >= S[l])
        {
            ++comparisons;
            l++;
        }
        while (r >= l && S[r] >= pivot)
        {
            ++comparisons;
            r--;
        }
        if (l < r)
        {
            dataMoves += 3;
            std::swap(S[l], S[r]);
        }                                                               // both
elements found
    }                                                                       //
until indices cross
    dataMoves += 3;
    std::swap(S[l], S[b]);                                                  //
store pivot at l
    quickSortStep(S, a, l - 1, comparisons, dataMoves);                     //
recur on both sides
    quickSortStep(S, l + 1, b, comparisons, dataMoves);
  }

void quickSort(std::vector<int>& S, long long &comparisons, long long
&dataMoves)
{
    if (S.size() <= 1) return;                                              //
already sorted
        quickSortStep(S, 0, S.size() - 1, comparisons, dataMoves);          //
call sort utility
}


vector<int> generateVector(int n, int choice)
```

```cpp
{
    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;
            default:
                vec.push_back(rand() % (1000 * n));
                break;
        }
    }

    return vec;
}

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

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

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

    if (printVec)
        printVector(vec);

    quickSort(vec, comparisons, dataMoves);

    if (printVec)
        printVector(vec);
    cout << "Comparisons:\t" << comparisons << endl;
    cout << "Data moves:\t" << dataMoves << endl;

    cout << endl;
}


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

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

    cout << "Modified by: 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:     36

Descending list:
9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9
Comparisons:    45
Data moves:     36

Random list:
6815 7666 963 2794 1349 2275 8206 7998 6202 9135
963 1349 2275 2794 6202 6815 7666 7998 8206 9135
Comparisons:    27
Data moves:     34

Large random list:
Comparisons:    699
Data moves:     532

Modified by: Nero Li
```

Exercise 2 -- need to submit source code and I/O

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

Source code below:
exercise_2.cpp:

```
/*  Program: PA_4_exercise_2
    Author: Nero Li
    Class: CSCI 230
    Date: 03/22/2022
    Description:
        Download a small data file small1k.txt containing 1,000 values
        in random order. In addition, download the large data file
        large100k.txt containing 100,000 values in random order. Sort
        the two data files using Quick-Sort-3 ("median of three" as
        pivot so you need to modify quick sort from exercise 1). For
        each value in the file, you must create a <key, value> pair and
        then sort a list of <key, value> pairs. You will have to sort
        two different lists of <integer, string> pairs and <string,
        integer> pairs. For instance, you need to create <1234, “1234”>
        pair and <”1234”, 1234> pair for an input value of 1234 from
        the file. Output all relevant information below for each input
        data file. There should be 4 sets of output for the two data
        files.

    I certify that the code below is my own work.

    Exception(s): N/A

*/

#include <iostream>
#include <fstream>
#include <vector>
#include <chrono>
#include <string>

using namespace std;

class compForIntKey
{
public:
    bool operator()(pair<int, string> a, pair<int, string> b) const
    {
        return a.first >= b.first;
    }
};

class compForStrKey
{
public:
    bool operator()(pair<string, int> a, pair<string, int> b) const
    {
        return a.first >= b.first;
    }
};

template <typename K, typename V, typename C>
void quickSortStep(std::vector<pair<K, V>>& S, int a, int b, const C
&compare, long long &comparisons, long long &dataMoves)
{
    if (a >= b) return;                                                 //
0 or 1 left? done
    int l = a;                                                          //
left edge
    int r = b - 1;                                                      //
right edge
    // find median of S[a], S[(a + b) / 2], S[b] and move into last spot
    K medOf3[3] = {S[a].first, S[(a + b) / 2].first, S[b].first};
    K result;

    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < i; ++j)
        {
            if (medOf3[i] < medOf3[j])
            {
                K temp = medOf3[j];
                medOf3[j] = medOf3[i];
                medOf3[i] = temp;
            }
        }
    }

    if (S[a].first == result)
    {
        dataMoves += 3;
        std::swap(S[a], S[b]);
    }
    else if (S[(a + b) / 2].first == result)
    {
        dataMoves += 3;
        std::swap(S[(a + b) / 2], S[b]);
    }

    ++dataMoves;
    auto pivot = S[b];
    while (l <= r)
    {
        while (l <= r && compare(pivot, S[l]))
        {
            ++comparisons;
            l++;
        }
        while (r >= l && compare(S[r], pivot))
        {
            ++comparisons;
            r--;
        }
        if (l < r)
        {
            dataMoves += 3;
            std::swap(S[l], S[r]);
        }                                                 // both
elements found
    }                                                          //
until indices cross
    dataMoves += 3;
    std::swap(S[l], S[b]);                                     //
store pivot at l
    quickSortStep<K, V, C>(S, a, l - 1, compare, comparisons,
dataMoves);                     // recur on both sides
    quickSortStep<K, V, C>(S, l + 1, b, compare, comparisons, dataMoves);
  }

template <typename K, typename V, typename C>
void quickSort(std::vector<pair<K, V>>& S, const C &compare, long long
&comparisons, long long &dataMoves)
{
    if (S.size() <= 1) return;                                 //
already sorted
        quickSortStep<K, V, C>(S, 0, S.size() - 1, compare, comparisons,
dataMoves);        // call sort utility
}
```

```
void testIntKey(string str)
{
    vector<pair<int, string>> vec;
    ifstream fin;
    fin.open(str, ios::binary);

    if(!fin)
        return;
    else
        cout << "For \"" << str << "\":\n";

    while (!fin.eof())
    {
        pair<int, string> newPair;
        int n;

        fin >> n;
        newPair.first = n;
        newPair.second = to_string(n);
        vec.push_back(newPair);
    }

    cout << "- Number of Values:\t\t" << vec.size() << endl;
    cout << "- Key Data Type:\t\tInteger\n";
    long long comparisons = 0;
    long long dataMoves = 0;
    compForIntKey compare;

    auto start = chrono::high_resolution_clock::now();
    quickSort<int, string, compForIntKey>(vec, compare, comparisons, 
dataMoves);
    auto end = chrono::high_resolution_clock::now();
    cout << "- Number of Key Compares:\t" << comparisons << endl;
    cout << "- Number of Data Moves:\t\t" << dataMoves << endl;
    cout << "- Time:\t\t\t\t" << 
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() * 
(double)1e-6) << " ms" << endl;

    cout << "- First 5 Entries:\n";
    for(int i = 0; i < 5; ++i)
        cout << "\t<" << vec[i].first << ", \"" << vec[i].second << 
"\">\n";
    cout << "- Last 5 Entries:\n";
    for(int i = vec.size() - 6; i < vec.size(); ++i)
```

```
        cout << "\t<" << vec[i].first << ", \"" << vec[i].second <<
"\">\n";

    cout << endl;
}

void testStrKey(string str)
{
    vector<pair<string, int>> vec;
    ifstream fin;
    fin.open(str, ios::binary);

    if(!fin)
        return;
    else
        cout << "For \"" << str << "\":\n";

    while (!fin.eof())
    {
        pair<string, int> newPair;
        int n;

        fin >> n;
        newPair.second = n;
        newPair.first = to_string(n);
        vec.push_back(newPair);
    }

    cout << "- Number of Values:\t\t" << vec.size() << endl;
    cout << "- Key Data Type:\t\tString\n";
    long long comparisons = 0;
    long long dataMoves = 0;
    compForStrKey compare;

    auto start = chrono::high_resolution_clock::now();
    quickSort<string, int, compForStrKey>(vec, compare, comparisons,
dataMoves);
    auto end = chrono::high_resolution_clock::now();
    cout << "- Number of Key Compares:\t" << comparisons << endl;
    cout << "- Number of Data Moves:\t\t" << dataMoves << endl;
    cout << "- Time:\t\t\t\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << "- First 5 Entries:\n";
    for(int i = 0; i < 5; ++i)
        cout << "\t<\"" << vec[i].first << "\", " << vec[i].second <<
">\n";
    cout << "- Last 5 Entries:\n";
    for(int i = vec.size() - 6; i < vec.size(); ++i)
        cout << "\t<\"" << vec[i].first << "\", " << vec[i].second <<
">\n";

    cout << endl;
}

int main()
{
    testIntKey("small1k.txt");
    testStrKey("small1k.txt");
    testIntKey("large100k.txt");
    testStrKey("large100k.txt");

    cout << "Modified by: Nero Li\n";

    return 0;
}
```

Input/output below:

```
For "small1k.txt":
- Number of Values:               1000
- Key Data Type:                  Integer
- Number of Key Compares:         11479
- Number of Data Moves:           7776
- Time:                           1.0003 ms
- First 5 Entries:
        <7, "7">
        <11, "11">
        <15, "15">
        <39, "39">
        <59, "59">
- Last 5 Entries:
        <8155, "8155">
        <8163, "8163">
        <8167, "8167">
        <8175, "8175">
        <8183, "8183">
        <8191, "8191">

For "small1k.txt":
- Number of Values:               1000
- Key Data Type:                String
- Number of Key Compares:       11245
- Number of Data Moves:         7761
- Time:                         1.0011 ms
- First 5 Entries:
        <"103", 103>
        <"1035", 1035>
        <"1047", 1047>
        <"1055", 1055>
        <"1063", 1063>
- Last 5 Entries:
        <"95", 95>
        <"955", 955>
        <"959", 959>
        <"987", 987>
        <"99", 99>
        <"995", 995>

For "large100k.txt":
- Number of Values:             100001
- Key Data Type:                Integer
- Number of Key Compares:       2005290
- Number of Data Moves:         1240491
- Time:                         106.097 ms
- First 5 Entries:
        <1, "1">
        <2, "2">
        <3, "3">
        <4, "4">
        <5, "5">
- Last 5 Entries:
        <99995, "99995">
        <99996, "99996">
        <99997, "99997">
        <99998, "99998">
        <99999, "99999">
        <100000, "100000">

For "large100k.txt":
- Number of Values:             100001
- Key Data Type:                String
- Number of Key Compares:       1981199
- Number of Data Moves:         1239642
- Time:                         151.144 ms
- First 5 Entries:
        <"1", 1>
        <"10", 10>
        <"100", 100>
        <"1000", 1000>
        <"10000", 10000>
- Last 5 Entries:
        <"99994", 99994>
        <"99995", 99995>
        <"99996", 99996>
        <"99997", 99997>
        <"99998", 99998>
        <"99999", 99999>
```

```
Modified by: Nero Li
```

Answer for Question 1:

It seems reasonable since their gap should be shown as multiple of 10. However, to the actual time for comparing, it depends on the modification for our current vector. If we got the comparisons count between 10 to 50 for 10 random values and 100 to 500 for 100 random values, then it is reasonable whatever the pivot choice method we are using in our code.

Answer for Question 2:

The collected run times between small file and large file, whatever the key type is integer or string, is reasonable since it showed the gap that is a multiple of 100 and add some more milliseconds due to the logN function.

For different key value, it also seems reasonable since when we use string as our key type, it will increase the running time due to the string comparison need to check out character one by one.

Extra Credit

Source code below:

extra_credit.cpp:

```
/*  Program: PA_4_extra_credit
    Author: Nero Li
    Class: CSCI 230
    Date: 03/22/2022
    Description:
        Perform exercise 2 and collect data for either merge sort, heap
        sort, or Shell sort (pick one).

    I certify that the code below is my own work.

    Exception(s): N/A
```

```cpp
*/

#include <iostream>
#include <fstream>
#include <vector>
#include <chrono>
#include <string>

using namespace std;

class compForIntKey
{
public:
    bool operator()(pair<int, string> a, pair<int, string> b) const
    {
        return a.first < b.first;
    }
};

class compForStrKey
{
public:
    bool operator()(pair<string, int> a, pair<string, int> b) const
    {
        return a.first < b.first;
    }
};

template <typename K, typename V, typename C>
void merge(vector<pair<K, V>> &vec, vector<pair<K, V>> a, vector<pair<K,
V>> b, const C &compare, long long &comparisons, long long &dataMoves)
{
    int i = 0, j = 0;

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

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

template <typename K, typename V, typename C>
void mergeSort(vector<pair<K, V>> &vec, const C &compare, long long
&comparisons, long long &dataMoves)
{
    if (vec.size() <= 1)
        return;

    vector<pair<K, V>> a;
    vector<pair<K, V>> b;

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

    mergeSort<K, V, C>(a, compare, comparisons, dataMoves);
    mergeSort<K, V, C>(b, compare, comparisons, dataMoves);
    vec.clear();
    merge<K, V, C>(vec, a, b, compare, comparisons, dataMoves);
}

void testIntKey(string str)
{
    vector<pair<int, string>> vec;
    ifstream fin;
    fin.open(str, ios::binary);

    if(!fin)
        return;
    else
        cout << "For \"" << str << "\":\n";
    while (!fin.eof())
    {
        pair<int, string> newPair;
        int n;

        fin >> n;
        newPair.first = n;
        newPair.second = to_string(n);
        vec.push_back(newPair);
    }

    cout << "- Number of Values:\t\t" << vec.size() << endl;
    cout << "- Key Data Type:\t\tInteger\n";
    long long comparisons = 0;
    long long dataMoves = 0;
    compForIntKey compare;

    auto start = chrono::high_resolution_clock::now();
    mergeSort<int, string, compForIntKey>(vec, compare, comparisons,
dataMoves);
    auto end = chrono::high_resolution_clock::now();
    cout << "- Number of Key Compares:\t" << comparisons << endl;
    cout << "- Number of Data Moves:\t\t" << dataMoves << endl;
    cout << "- Time:\t\t\t\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << "- First 5 Entries:\n";
    for(int i = 0; i < 5; ++i)
        cout << "\t<" << vec[i].first << ", \"" << vec[i].second <<
"\">\n";
    cout << "- Last 5 Entries:\n";
    for(int i = vec.size() - 6; i < vec.size(); ++i)
        cout << "\t<" << vec[i].first << ", \"" << vec[i].second <<
"\">\n";

    cout << endl;
}

void testStrKey(string str)
{
    vector<pair<string, int>> vec;
    ifstream fin;
    fin.open(str, ios::binary);
    if(!fin)
        return;
    else
        cout << "For \"" << str << "\":\n";

    while (!fin.eof())
    {
        pair<string, int> newPair;
        int n;

        fin >> n;
        newPair.second = n;
        newPair.first = to_string(n);
        vec.push_back(newPair);
    }

    cout << "- Number of Values:\t\t" << vec.size() << endl;
    cout << "- Key Data Type:\t\tString\n";
    long long comparisons = 0;
    long long dataMoves = 0;
    compForStrKey compare;

    auto start = chrono::high_resolution_clock::now();
    mergeSort<string, int, compForStrKey>(vec, compare, comparisons,
dataMoves);
    auto end = chrono::high_resolution_clock::now();
    cout << "- Number of Key Compares:\t" << comparisons << endl;
    cout << "- Number of Data Moves:\t\t" << dataMoves << endl;
    cout << "- Time:\t\t\t\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << "- First 5 Entries:\n";
    for(int i = 0; i < 5; ++i)
        cout << "\t<\"" << vec[i].first << "\", " << vec[i].second <<
">\n";
    cout << "- Last 5 Entries:\n";
    for(int i = vec.size() - 6; i < vec.size(); ++i)
        cout << "\t<\"" << vec[i].first << "\", " << vec[i].second <<
">\n";

    cout << endl;
}

int main()
```

```
{
    testIntKey("small1k.txt");
    testStrKey("small1k.txt");
    testIntKey("large100k.txt");
    testStrKey("large100k.txt");

    cout << "Modified by: Nero Li\n";

    return 0;
}
```

Input/output below:

```
For "small1k.txt":
- Number of Values:                1000
- Key Data Type:                   Integer
- Number of Key Compares:          8686
- Number of Data Moves:            19952
- Time:                            13.0121 ms
- First 5 Entries:
        <7, "7">
        <11, "11">
        <15, "15">
        <39, "39">
        <59, "59">
- Last 5 Entries:
        <8155, "8155">
        <8163, "8163">
        <8167, "8167">
        <8175, "8175">
        <8183, "8183">
        <8191, "8191">

For "small1k.txt":
- Number of Values:                1000
- Key Data Type:                   String
- Number of Key Compares:          8695
- Number of Data Moves:            19952
- Time:                            36.0341 ms
- First 5 Entries:
        <"103", 103>
        <"1035", 1035>
        <"1047", 1047>
        <"1055", 1055>
        <"1063", 1063>
- Last 5 Entries:
        <"95", 95>
        <"955", 955>
        <"959", 959>
        <"987", 987>
        <"99", 99>
        <"995", 995>

For "large100k.txt":
- Number of Values:             100001
- Key Data Type:                Integer
- Number of Key Compares:       1536326
- Number of Data Moves:         3337892
- Time:                         1356.37 ms
- First 5 Entries:
        <1, "1">
        <2, "2">
        <3, "3">
        <4, "4">
        <5, "5">
- Last 5 Entries:
        <99995, "99995">
        <99996, "99996">
        <99997, "99997">
        <99998, "99998">
        <99999, "99999">
        <100000, "100000">

For "large100k.txt":
- Number of Values:             100001
- Key Data Type:                String
- Number of Key Compares:       1536350
- Number of Data Moves:         3337892
- Time:                         1285.42 ms
- First 5 Entries:
        <"1", 1>
        <"10", 10>
        <"100", 100>
        <"1000", 1000>
        <"10000", 10000>
- Last 5 Entries:
        <"99994", 99994>
        <"99995", 99995>
        <"99996", 99996>
        <"99997", 99997>
        <"99998", 99998>
        <"99999", 99999>

Modified by: Nero Li
```