Evan Nguyen
Joseph Guzman
CECS 275 - 03/11
Group #12

Lab #3

Screenshots for Problem 1:

C: > Users > jayge > Lab3_1.cpp > outToFile(vector<int>, int, int, float)

```cpp
/*
 * Answer to #1 on Lab 3
 * CECS 275 - Spring 2022
 * @author Joseph Guzman
 * @author Evan Nguyen
 * @version 1.0.0
 *
 */

#include <iostream>
#include <vector>
#include <fstream>
#include <iomanip>
#include <cmath>

using namespace std;

// vector<int> read_data - takes read_data.txt as an argument and returns the data as a vector<int>
// vector<int> sort_data - takes a vector<int> and int vecSize as argument, sorts it, returns new sorted vector<int>
// int max - takes a sorted vector<int> as an argument and returns the maximum value
// int min - takes a sorted vector<int> as an argument and returns the minimum value
// float avg - takes a vector<int> as an argument and returns the average
// vector<int> outToFile - takes sorted vector<int>, int min, int max, float avg and outputs to file
// void swap - takes reference arguments a & b and swaps them. use for sorting

vector<int> read_data(string inputFile);
vector<int> sort_data(vector<int> rawData, int vecSize);
int max_val(vector<int> sortedData);
int min_val(vector<int> sortedData);
float avg(vector<int> sortedData, int vecSize);
void outToFile(vector<int> sortedData, int minValue, int maxValue, float avg);
void swap(int &a, int&b);

int main(){

    string file = "data.txt";
    int vecSize = 0;

    vector<int> rawData = read_data(file);

    // get the vector size to use in other functions
    for (int h : rawData){
```

```
43          vecSize++;
44      }
45
46      vector<int> sortedData = sort_data(rawData, vecSize);
47
48      int minValue = min_val(sortedData);
49      int maxValue = max_val(sortedData);
50
51      float avgValue = avg(sortedData, vecSize);
52      outToFile(sortedData, minValue, maxValue, avgValue);
53
54      return 0;
55  }
56
57  // swap function for sorting
58  void swap(int &a, int &b) {
59      int temp = a;
60      a = b;
61      b = temp;
62  }
63
64  // read data.txt and output it cleanly to a vector int
65  vector<int> read_data(string inputFile) {
66      vector<int> number_List;
67      string word, temp_Word;
68      int temp_Num, int1, newInt, store;
69      int temp = 0;
70
71      ifstream inFile;
72      inFile.open(inputFile);
73
74      if (!inFile){
75          cout << "File Not Found!" << endl;
76      }
77      else{
78          cout << "File Found!" << endl;
79      }
80      //int i = 0;
81      while(inFile >> word){
82          //For loop that removes commas from the string
83          for (int i = 0; i < word.length(); i++){
```

```
        for (int i = 0; i < word.length(); i++){
            if (word[i] == ',')
            {
                word[i] = ' ';
            }
            temp_Word += word[i]; // temp_Word holds our number list with spaces instead of commas
        }
    }

    cout << temp_Word;


    // convert the word to ascii, modulus by 10 and get the integer value. pushback into vector
    for (int i = 0; i < temp_Word.length(); i++){
        if (temp_Word[i] != ' ') { // ignore space
            int1 = (temp_Word[i] - '0') % 10;
            temp *= 10;
            newInt = int1 + temp;
            //cout << "newInt: " << newInt << endl;
            temp = int1;
        }
        else {
            store = newInt;
            number_List.push_back(store);
            //cout << "store: " << store << endl;
            int1 = 0;
            temp = 0;
        }
    }
    store = newInt;
    number_List.push_back(store);

    //cout << "store at end: " << store << endl;

    for (int i : number_List){
        cout << i << " ";
    }
    return number_List;
}

// sorts rawData and returns a sorted vector
```

```cpp
vector<int> sort_data(vector<int> rawData, int vecSize){
    int minIndex, minValue;

    // selection sort
    for (int start = 0; start < (vecSize-1); start++){
        minIndex = start;
        minValue = rawData[start];
        for(int index = start + 1; index < vecSize; index++){
            if (rawData[index] < minValue){
                minValue = rawData[index];
                minIndex = index;
            }
        }
        swap(rawData[minIndex], rawData[start]);
    }

    // check if array sorted
    cout << endl << endl << "Sorted from Min to Max" << endl;
    for (int j : rawData){
        cout << j << " ";
    }
    cout << endl;

    return rawData;
}

// min value is the first element in a sorted vector
int min_val(vector<int> sortedData){

    cout << "\nSorted Data Min Value: " << sortedData[0] << endl;

    return sortedData[0];
}

// max value is the last element in a sorted vector
int max_val(vector<int> sortedData){

    cout << "\nSorted Data Max Value: " << sortedData[sortedData.size()-1] << endl;

    return sortedData[sortedData.size()-1];
}
```

```
}

// compute the average of the vector
float avg(vector<int> sortedData, int vecSize){
    double sum = 0.0;
    double average_2pt = 0.0;

    for (int i : sortedData){
        sum += i;
    }

    // round to 2 decimal points
    average_2pt = ceil((sum / vecSize) * 100.0) / 100.0;
    return average_2pt;
}

// output to file
void outToFile(vector<int> sortedData, int min, int max, float avg){
    vector<int> count;
    vector<int> unique;
    int value;
    int freq = 0;
    int unique_Val;
    int range = sortedData.size();
    int numS = 0;
    cout << '\n';

    ofstream outputFile("frequency.txt");
    for (int i = 0; i < range; i++){
        unique_Val = sortedData[i];
        unique.push_back(unique_Val); // store unique values
        int j = 0;

        cout << unique_Val;

        outputFile << unique_Val;
        // check unique values and determine if the next value is == to it.
        while (sortedData[i+j]==unique_Val){
            //cout << "sortedData[" << i << "+" << j << "]: " << sortedData[i+j] << " unique_Val: " << unique_Val << endl;
            j++;
            freq++;
            //cout << "Unique_Val: " << unique_Val << "Frequency: " << freq << endl;
```

```
        }

        cout << ":";
        outputFile << ":";

        // append *'s
        for (int i=0; i < freq; i++){
            if (freq != 0){
                cout <<"*";
                numS++;
                outputFile << "*";
            }
            else {
                break;
            }
        }

        // set teh width to accomodate 2 digit numbers
        if (unique_Val < 10) {
            cout << setw(9-numS);
            outputFile << setw(9-numS);
        } else {
            cout << setw(8-numS);
            outputFile << setw(8-numS);
        }

        cout << "(" << freq << ")" << endl;
        outputFile << " " << "(" << freq << ")" << endl;

        if (sortedData[i+j]!= unique_Val){
            i = i+j-1;
        }

        for (int c = 0; c < unique.size(); c++){
            if (((unique[c] + 1) != unique[c+1]))
                if (unique[c] + 1 > unique_Val && ((unique[c] + 1) < sortedData[i+1]))
                    cout << unique[c] + 1 << ":       (0)" << endl;
                    outputFile << unique[c] + 1 << ":        (0)" << endl;
                    //cout << unique[c] + 1 << ":        (0)" << endl;
        }
```

```
C:\Users\jayge>g++ Lab3_1.cpp

C:\Users\jayge>a
File Found!
36 14 29 28 31 7 33 4 49 16 45 36 15 36 44 34 4 19 3 31 47 33 2 19 44 4 26 23 28 32 30 47 18 2 22 1
5 43 30 28 45 5 24 3 5 6 6 6 9 27 31 34 4136 14 29 28 31 7 33 4 49 16 45 36 15 36 44 34 4 19 3 31 4
 47 7 37 39 33 6 11 26 9 10 6 9 10 48 43 45 43 30 28 45 5 24 3 5 6 6 6 9 27 31 34 41

Sorted from Min to Max
1 2 2 3 3 4 4 4 4 4 5 5 6 6 6 6 6 6 6 7 7 8 9 9 9 10 10 10 11 11 12 12 12 14 14 15 16 16 16 16 18 1
 43 43 43 44 44 44 45 45 45 47 47 47 48 49

Sorted Data Min Value: 1

Sorted Data Max Value: 49
```

```
246
247         numS = 0;
248         freq = 0;
249
250     }
251
252     outputFile << "The maximum is " << max << "." << endl;
253     outputFile << "The minimum is " << min << "." << endl;
254     outputFile << "The average is " << avg <<". (Format with 2 decimal places)";
255     outputFile.close();
256 }
257
```

```
C:\Users\jayge>a
File Found!
36 14 29 28 31 7 33 4 49 16 45 36 15 36 44 34 4 19 3 31 47 33 2 19 44 4 26 23 28 32 30 47 18 2 22 18 31 16 22 16 4 32 12 44 10 11 41 27 6 8 38 26 16 26 43 1 27 12 12 14 30 4 34 6 35 23
9 16 45 36 15 36 44 34 4 19 3 31 47 33 2 19 44 4 26 23 28 32 30 47 18 2 22 18 31 16 22 16 4 32 12 44 10 11 41 27 6 8 38 26 16 26 43 1 27 12 12 14 30 4 34 6 35 23 47 7 37 39 33 6 11 26 9

Sorted from Min to Max
1 2 2 3 3 4 4 4 4 4 5 5 6 6 6 6 6 6 7 7 8 9 9 9 10 10 10 11 11 12 12 12 14 14 15 16 16 16 16 18 18 19 19 22 22 23 23 24 26 26 26 26 27 27 27 28 28 28 29 30 30 30 31 31 31 31 32 32 33

Sorted Data Min Value: 1

Sorted Data Max Value: 49
```

```
1:*       (1)
2:**      (2)
3:**      (2)
4:*****   (5)
5:**      (2)
6:******* (7)
7:**      (2)
8:*       (1)
9:***     (3)
10:***    (3)
11:**     (2)
12:***    (3)
13:       (0)
14:**     (2)
15:*      (1)
16:****   (4)
17:       (0)
18:**     (2)
19:**     (2)
20:       (0)
22:**     (2)
23:**     (2)
24:*      (1)
25:       (0)
26:****   (4)
27:***    (3)
28:***    (3)
29:*      (1)
30:***    (3)
31:****   (4)
32:**     (2)
33:***    (3)
34:***    (3)
35:*      (1)
36:***    (3)
37:*      (1)
38:*      (1)
```

```
39:*      (1)
40:       (0)
41:**     (2)
42:       (0)
43:***    (3)
44:***    (3)
45:***    (3)
46:       (0)
47:***    (3)
48:*      (1)
49:*      (1)
50:       (0)

C:\Users\jayge>
```

C: > Users > jayge > data.txt

```
36, 14, 29, 28, 31, 7, 33, 4, 49, 16, 45, 36, 15, 36, 44, 34, 4, 19, 3, 31, 47, 33, 2, 19, 44, 4,
26, 23, 28, 32, 30, 47, 18, 2, 22, 18, 31, 16, 22, 16, 4, 32, 12, 44, 10, 11, 41, 27, 6, 8, 38, 26,
16, 26, 43, 1, 27, 12, 12, 14, 30, 4, 34, 6, 35, 23, 47, 7, 37, 39, 33, 6, 11, 26, 9, 10, 6, 9, 10,
48, 43, 45, 43, 30, 28, 45, 5, 24, 3, 5, 6, 6, 6, 9, 27, 31, 34, 41
```

C: > Users > jayge > Frequency.txt

```
1:*       (1)
2:**      (2)
3:**      (2)
4:*****   (5)
5:**      (2)
6:******* (7)
7:**      (2)
8:*       (1)
9:***     (3)
10:***    (3)
11:**     (2)
12:***    (3)
14:**     (2)
15:*      (1)
16:****   (4)
18:**     (2)
19:**     (2)
22:**     (2)
23:**     (2)
24:*      (1)
26:****   (4)
27:***    (3)
28:***    (3)
29:*      (1)
30:***    (3)
31:****   (4)
32:**     (2)
33:***    (3)
34:***    (3)
35:*      (1)
36:***    (3)
37:*      (1)
38:*      (1)
39:*      (1)
41:**     (2)
43:***    (3)
44:***    (3)
45:***    (3)
47:***    (3)
48:*      (1)
49:*      (1)
The maximum is 49.
The minimum is 1.
The average is 23.12. (Format with 2 decimal places)
```

Screenshots for Problem 2

C: > Users > jayge > Lab3_2.cpp > main()

```cpp
/*
 * Answer to #2 on Lab 3
 * CECS 275 - Spring 2022
 * @author Joseph Guzman
 * @author Evan Nguyen
 * @version 1.0.0
 *
 */
#include <iostream>

using namespace std;

void reverse(char s[]);
void concat(const char a[], const char b[], char result[], int result_maxlength);

int main(){
    cout << "\n*******************TESTING**********************\n";
    // FUNCTION 1: reverse
    // change s[] to get varying outputs
    cout << "FUNCTION 1: reverse \n";
    char s[] = "Hello, this is Evan. Hello this is Joseph";
    reverse(s);

    cout << "-------------------------------------------------\n";

    // FUNCTION 2: concat
    // result_maxlength = 5. starting max_length
    // max length 10 and 20 are outputted, as seein in the sample run
    cout << "FUNCTION 2: concat \n";
    int result_maxlength = 5;
    const char a[] = "chicken";
    const char b[] = "waffle";
    char result[result_maxlength]; // buffer result
    concat(a, b, result, result_maxlength);
    cout << "\n************************************************\n\n";
    return 0;
}

void reverse(char s[]){

    // create pointer to char
    char *ptr = s;
    int i = 0;
    int sizeOf = 0;

    // iterate through the address and checks if it returns an item
    bool isDone = false;
    while(isDone == false){
        if (ptr[i]) {
            sizeOf++; // still in the array
        } else {
```

```
44      int sizeOf = 0;
45
46      // iterate through the address and checks if it returns an item
47      bool isDone = false;
48      while(isDone == false){
49          if (ptr[i]) {
50              sizeOf++; // still in the array
51          } else {
52              isDone = true; // not in the array, flag isDone
53          }
54          i++;
55      }
56
57      // I think its better to point rather than access directly,
58      // hence why i created char *ptr = s rather than
59      // doing *(s + 1) and so forth.
60      cout << "\"";
61      for (int i = 0; i <= sizeOf; i++){
62          cout << ptr[i];
63      }
64      cout << "\"";
65      cout << " becomes ";
66      cout <<"\"";
67
68      // define new string with new size
69      char newStr[sizeOf];
70      for (int i = 0; i <= sizeOf; i++){
71          newStr[i] = ptr[sizeOf - i];
72          cout << newStr[i];
73      }
74      cout <<"\".";
75      cout << endl;
76
77  }
78
79  void concat(const char a[], const char b[], char result[], int result_maxlength){
80
81      // create pointer to constants
82      const char *ptrA = a;
83      const char *ptrB = b;
84
85      // get the size of a -> add to new char array
86      int i = 0;
87      int sizeOfA = 0;
88      bool isDone = false;
89      while(isDone == false){
90          if (ptrA[i]) {
91              sizeOfA++; // still in the array
```

```
92          } else {
93              isDone = true; // not in the array, flag isDone
94          }
95          i++;
96      }
97
98      // get the size of b -> add to new char array. reset flags
99      i = 0;
100     int sizeOfB = 0;
101     isDone = false;
102     while(isDone == false){
103         if (ptrB[i]) {
104             sizeOfB++; // still in the array
105         } else {
106             isDone = true; // not in the array, flag isDone
107         }
108         i++;
109     }
110
111     int sizeBoth = sizeOfA + sizeOfB;
112     char buffer[sizeBoth];
113     char *ptr = buffer; // pointer to buffer
114
115     // add array a and b via dereference. -1 to remove null character
116     for (int j = 0; j <= sizeOfA-1; j++){
117         *ptr++ = a[j];
118     }
119     for (int k = 0; k <= sizeOfB-1; k++){
120         *ptr++ = b[k];
121     }
122
123     // point to original char a[] and char[b]
124     cout << "If char a[] = \"";
125     for (int i = 0; i <= sizeOfA; i++){
126         cout << ptrA[i];
127     }
128
129     cout << "\" and char b[] = \"";
130     for (int i = 0; i <= sizeOfB; i++){
131         cout << ptrB[i];
132     }
133
134     // -Output the buffer
135     // If result_maxlength is greater than sizeBoth,
136     // just output the entire string. Ignore extraneous characters.
137     // Accessing out of bound array elements results in weird characters
138     cout << "\"\nthen:";
139
140     // this code block creates  test
141     // cases as seen in the sample run.
142     int *ptrR = &result_maxlength;
```

```
143     int testCase = 0;
144     while (testCase != 3){
145 
146         int outLength = *ptrR;
147         if (*ptrR > sizeBoth){
148             outLength = sizeBoth;
149         }
150 
151         cout << "\nmax_length = " << *ptrR << " ---> \"";
152         for (int o = 0; o < outLength; o++){
153             cout << buffer[o];
154             // point to result and change it based on these conditions
155         }
156 
157         cout << "\"";
158         *ptrR *= 2;
159         testCase++;
160     }
161 }
162 
```

```
C:\Users\jayge>g++ Lab3_2.cpp

C:\Users\jayge>a

**********TESTING**********
FUNCTION 1: reverse
"Hello, this is Evan. Hello this is Joseph" becomes "hpesoJ si siht olleH .navE si siht ,olleH".
-----------------------------------------------
FUNCTION 2: concat
If char a[] = "chicken" and char b[] = "waffle"
then:
max_length = 5 ---> "chick"
max_length = 10 ---> "chickenwaf"
max_length = 20 ---> "chickenwaffle"
***********************************************


C:\Users\jayge>
```