CSCI 2270 – Data Structures Spring 2020

Instructors: Zagrodzki, Ashraf, Trivedi

Assignment 1

Due Sunday, January 26, 6PM

C++ FUNDAMENTALS

OBJECTIVES

1. Read-in command line arguments
2. Read a file
3. Loop through an array
4. Split a string
5. Create an array of struct types
6. Read and Write to a file

Instructions

Write code to complete the **Problems 1 and 2**. Implement each of the problems as separate programs. To receive credit for your code, you will need to paste your solution into **Moodle's autograder (CodeRunner)**. Your code needs to compile and pass the given test cases in order to receive points. Note that the autograder submissions are divided into separate parts, so that you will be required to test the individual functions prior to testing the entire program.

Problem 1

Overview: You will write a program that reads up to 100 numbers from a file. As you read the numbers, insert them into an array in descending order.

Specifications:

1A. Write a function called **insertIntoSortedArray**.

i. It should take **three** arguments -

a. myArray[ ] : *sorted* array that should be able to hold at most 100 floats.
b. numEntries : the number of elements inserted so far.
c. newValue : the incoming value to be inserted into the sorted array (i.e. myArray[]).

ii. The **insertIntoSortedArray** function should return a count of the elements inserted so far (i.e. the current size of the array)

The function header will appear as follows:

```
int insertIntoSortedArray(float myArray[], int numEntries, float newValue);
```

CSCI 2270 – Data Structures Spring 2020

Instructors: Zagrodzki, Ashraf, Trivedi

1B. Write a **complete program** to do the following **:**

i. Reading the file: Your program should take a single command line argument i.e. the name of the file where the numbers are present.

a. This file needs to be stored in the same directory as your program.
b. The file should store up to 100 numbers on separate lines. For testing you can use the file named *"**numbers.txt**"* on Moodle, or create your own if you prefer.

ii. In the main function:

a. Create an array of floats to store at most 100 floats.
b. Open the file that was passed via the command line
    i. If there is any error in opening the file then print the below statement *std::cout << "Failed to open the file." << std::endl;*
c. Use the **getline** function to read the integers one by one.
d. Store these integers in a sorted array by passing them to the **insertIntoSortedArray** function (you should use the code from part 1A).
e. Each time a new number is read, print out the entire array after insertion.

The Input and Output formats are shown below:

Testcase 1:
FileContents: arr.txt
1
6
2
12
5
Your Output:
1
6, 1
6, 2, 1
12, 6, 2, 1
12, 6, 5, 2, 1

CSCI 2270 – Data Structures Spring 2020

Instructors: Zagrodzki, Ashraf, Trivedi

Problem 2

Overview: In this question, you will write a program that:

1. Reads a **".csv"** file with up to 100 lines and columns containing information on national parks.
2. Stores the information in an array of structs.
3. Writes the lines into the **output .csv** file where the area of the park is within a range specified by the lower bound and upper bound.
4. Prints the content of the entire array.

Specifics:

Create an array that holds the **Park struct objects**. Use the following struct declaration:

```
struct Park{
   string parkname;
   string state;
   int area;
};
```

2A. Write a function named **addPark:**

i. The **addPark** function has the following declaration:

```
// length: Number of items currently stored in the array
void addPark(Park parks[], string parkname, string state, int area, int length);
```

ii. Instantiate a struct and store the **parkname, state**, and **area** values in it.

iii. Add the struct to the **parks** array.

2B. Write a function named **printList:**

i. The **printList** function has the following signature:

```
// length: Number of items in the array
void printList(const Park parks[], int length);
```

ii. Loop through the **parks** array.

iii. Print out each element of the **parks** array in the following format.

"<PARKNAME> [<STATE>] area: <AREA>" using the below cout statement

std::*cout << park.parkname <<" [" << park.state << "] area: "<< park.area << std::endl;*

Example:

"Acadia National Park [ME] area: 47390"

CSCI 2270 – Data Structures Spring 2020

Instructors: Zagrodzki, Ashraf, Trivedi

2C. Write a **complete program** which includes the following**:**

I. The **park** struct and the **addPark, printList** functions coded above.

II. A **main()** function defined as below:

1. Your **main()** should handle **four command line arguments**: the name of the **input** "**.csv" file**, the name of the **output** "**.csv" file**, a **lower bound of area** and **an upper bound of area** respectively**.**
2. Input and output files need to be stored in the same directory as your program.
3. Read from the input file, "***park.csv***" :
    a. Each line of the file can be read using **getline** function.
    b. Parse each line using **stringstream** and convert each entry into its appropriate data type. **parkname** should be a string, **state** should be a string, and **area** should be an integer. *(Hint: Use **stoi, stof** functions to convert from strings to numbers)*
    c. Call **addPark** function to update the **parks** array.
4. Call the **printList** function after the array has been filled with data.
5. Writing out to external files in C++ can be done very similarly as reading in. Instead of using an object of the *ifstream* class, as is done with input streams, use an object of the *ofstream* class. A "csv" stands for comma separated values. Write into **output ".csv"** file:
    a. Write the <parkname>, <state>, <area> of the parks, whose <area> >= **lower_bound** and <area> <= **upper_bound** (read from command line) into the **output ".csv"** file.
    b. You should not sort them while writing to output file. The relative order will be same as input file. Only those park with area within the bound should be written in the output file.
6. Make sure your program closes the output file when it is finished writing.

Check next page for sample input and output.

CSCI 2270 – Data Structures Spring 2020

Instructors: Zagrodzki, Ashraf, Trivedi

Sample Input and Output:

Testcase 1:

File Contents: data.csv

Acadia National Park, ME, 47390
Arches National Park, UT, 76519
Badlands National Park, SD, 242756
Big Bend National Park, TX, 801163
Biscayne National Park, FL, 172924

Your print Output:

Acadia National Park [ME] area: 47390
Arches National Park [UT] area: 76519
Badlands National Park [SD] area: 242756
Big Bend National Park [TX] area: 801163
Biscayne National Park [FL] area: 172924

Your output.csv file with area between 200000 and 900000 should contain the following:

Badlands National Park, SD, 242756
Big Bend National Park, TX, 801163