mmAY 2020 Assignment 5 [8 Marks]

Date / Time	19 November 2021 – 3 December 2021 23:59
Course	**[M1522.600] Computer Programming**
Instructor	Youngki Lee

- You can refer to the Internet or other materials to solve the assignment, but you ***SHOULD NOT*** discuss the question with anyone else and need to code **ALONE**.
- We will use the automated copy detector to check for potential plagiarism in the codes between the students. The copy checker is made to be very reliable so that it is highly likely to mark a pair of code as a copy even though two students quickly discuss the idea without looking at each other's code. Of course, we will evaluate the similarity of a pair compared to the overall similarity for the entire class.
- We will also inspect the codes manually. In case we doubt that the code may be written by someone else (outside of the class), we reserve the right to request an explanation about the code. We will ask detailed questions that can only be answered when the codes were written by yourself.
- If one of the above cases happens, you will get 0 marks for the assignment and may get a further penalty. Please understand that we apply these methods for the fairness of the assignment.
- Download and unzip “HW5.zip” file from the autolab. “HW5.zip” file contains skeleton codes for Question 1 and Question 2 (in the “problem1”, “problem2” directory) .
- Do not modify the overall directory structure after unzipping the file, and fill in the code in appropriate files. It is okay to add new directories or files if needed.
- When you submit, compress the “HW5” directory which contains “problem1” and “problem2” directories in a single zip file named “20XX-XXXXX.zip” (your student ID) and upload it to autolab as you submit the solution for HW1~HW4. Contact the TA if you are not sure how to submit. Double-check if your final zip file is properly submitted. You will get 0 marks for the wrong submission format.
- C / C++ Standard Library (including Standard Template Library) is allowed.
- CLion sometimes successfully completes the compilation even without including some header files. Nevertheless, you **MUST include all the necessary header files in the code** to make it successfully compile in the autolab server. Otherwise, you may not be able to get the points.
- Do not use any external libraries.

Contents

Submission Guidelines

1. You should submit your code on autolab.
2. After you extract the zip file, you must have a "HW5" directory. The submission directory structure should be as shown in the table below.
3. You can create additional directories or files in each problem directory. Make sure to update CMakeLists.txt to reflect your code structure changes.
4. You can add additional methods or classes, but do not remove or change signatures of existing methods.
5. Compress the "HW5" directory and name the file "20XX-XXXXX.zip" (your student ID).

Submission Directory Structure (Directories or Files can be added)

- Inside the "HW5" directory, there should be "problem1" and "problem2" directory.

Directory Structure of Question 1	Directory Structure of Question 2
problem1/ └── CMakeLists.txt └── main.cpp └── p1-1.cpp └── p1-2.cpp └── p1-3.cpp └── p1-4.cpp └── p1-5.cpp	problem2/ └── test/... └── CMakeLists.txt └── main.cpp └── CSI.cpp └── CSI.h └── TestHelper.cpp └── TestHelper.h

Question 1: C++ Basics Exercise[3 Marks]

For the sub-questions of Question 1, there will be NO partial points.

Question 1-1: Palindrome String [0.5 Marks]

Objective: Write a function that checks whether the given string is a palindrome or not.

Description:

- Palindrome string is a string that is the same when written forwards or backwards.

Target Function: `bool is_palindrome(string s)` (in p1-1.cpp)
Parameter: string s (0 <= length <= 200)
Return value: True if s is palindrome, false otherwise.

Input	Output
`s = “”`	`true`
`s = “ab”`	`false`
`s = “aaabbb”`	`false`
`s = “12321a12321”`	`true`

Question 1-2: Hamming Distance [0.5 Marks]

Objective: Write a function that calculates the hamming distance between two numbers.

Description:

- The Hamming distance between two integers is the number of positions at which the corresponding bits are different.
- For example, given 1 and 4, the binary format of 1 is “001” and the binary format of 4 is “100”. Then you can see that the leftmost bit and the rightmost bit are different. Therefore, the hamming distance is 2.

```
1   (0 0 0 1)
4   (0 1 0 0)
      ↑   ↑
```

Target Function: `int hamming_distance(int x, int y)` (in p1-2.cpp)
Parameter: integer `x`, `y` (0 <= `x`, `y` <= 2^{31}-1)

Return value: The hamming distance between `x` and `y`.

Input	Output
`x = 1, y = 4`	`2`
`x = 3, y = 1`	`1`
`x = 33, y = 15`	`4`

Question 1-3: Merge Arrays [0.5 Marks]

Objective: Write a function that merges two arrays that are sorted into a single array that is sorted.

Description:

- Two arrays, `arr1` and `arr2`, that are sorted in non-decreasing order are given. You should merge the two arrays into a single array in a non-decreasing order.
- The final sorted array should NOT be returned by the function, but instead be stored inside the first array, `arr1`. To accommodate this, `arr1` has a length of `len1` + `len2`, where the first `len1` elements denote the sorted elements that should be merged, and the last `len2` elements are set to 0 and should be ignored. `arr2` has a length of `len2`.

Target Function: `void merge_arrays(int* arr1, int len1, int* arr2, int len2)` (in p1-3.cpp)
Parameter:

- `arr1` and `arr2` are the two arrays that are sorted in non-decreasing order.
- `len1` + `len2` and `len2` are the lengths of `arr1` and `arr2`, respectively.
- 1 <= `len1`, `len2` <= 200

Return value: There is no return value. You should modify the `arr1`.

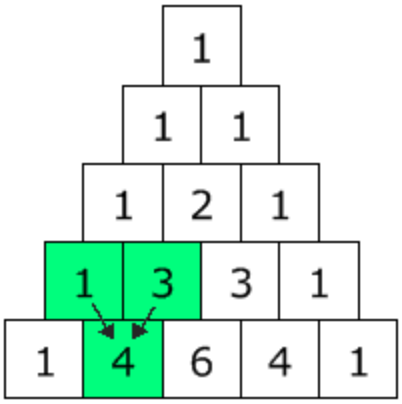
Input	Output
`arr1={1,2,3,0,0,0}, len1=3, arr2={4,5,6}, len2=3`	`arr1 modified to {1,2,3,4,5,6}`
`arr1={1,3,5,0,0}, len1=3, arr2={3,5}, len2=2`	`arr1 modified to {1,3,3,5,5}`
`arr1={-3,10,15,16,0,0,0}, len1=4, arr2={-99,-5,50}, len2=3`	`arr1 modified to {-99,-5,-3,10,15,16,50}`

Question 1-4: Pascal's Triangle [0.5 Marks]

Objective: Write a function that returns the N-th row of the Pascal's triangle

Description:

- Pascal's triangle is a triangular array of the binomial coefficients that arises in probability theory, combinatorics, and algebra. It is named after the French mathematician Blaise Pascal.
- For each row, the leftmost number and the rightmost number is always 1.
- Other numbers can be calculated by summing the two numbers directly above it as shown in the figure below.



Target Function: `int* pascal_triangle(int N)` (in p1-4.cpp)
Parameter:

- `N` indicates which row of the triangle to return. (1 <= N <= 20)

Return value: Array of length `N`. You should allocate the proper amount of memory space for the array.

Input	Output
`N = 1`	`{1}`
`N = 3`	`{1,2,1}`
`N = 5`	`{1,4,6,4,1}`

Question 1-5: School Cafeteria[1 Marks]

Objective: Write a function that checks whether you can pay every customer with correct change.

Description:

- You are an owner of the school cafeteria where only one menu, *bibimbap*, is available.
- Bibimbap costs $5 and the customer must pay in cash. Credit cards are not allowed.
- Customers are standing in a queue to buy from you, and order one at a time.
- Each customer will only buy one bibimbap and pay with either a $5, $10, or $20 bill.
- You must provide the correct change to each customer.
- You don't have any change in hand at first.

Target Function: `bool bibimbap_change(int* bills, int N)` (in p1-5.cpp)
Parameter:

- `bills` where `bills[i]` is the bill that the i-th customer pays. `bills[i]` is either 5, 10 or 20.
- `N` indicates the number of customers (= length of bills). (1 <= `N` <= 1000)

Return value: True if you can provide every customer with correct change, false otherwise.

Input	Output	Explanation
`bills = {5,5,5,10,20}`	`true`	`From the first 3 customers, you collect three $5 bills. From the fourth customer, you collect a $10 bill and give back a $5 bill. For the fifth customer, you give a $10 bill and a $5 bill.`
`bills = {5,5,10,10,20}`	`false`	`From the first two customers, you collect two $5 bills. For the next two customers, you collect two $10 bills and give back two $5 bills. For the last customer, you cannot give change of $15 back because you only have two $10 bills in hand.`
`bills = {10,10}`	`false`	`You cannot give change to the first customer since you don't have any change in hand at first.`

Question 2: Wi-Fi Signal Decoder [5 Marks]

Objective: Develop a Wi-Fi signal decoder to convert “Wi-Fi Channel State Information (CSI)” into human-interpretable amplitude values. Do not worry about the terms like CSI or amplitude; we will explain necessary details later.

Description: You are a research intern at professor Youngki’s lab, currently working on analyzing Wi-Fi signal patterns to recognize a user’s activity (such as walking, standing, sitting). You are asked to implement a small sub-component, i.e., a Wi-Fi signal decoder, which converts low-level “Wi-Fi Channel State Information (CSI)” into human-interpretable amplitude values.

Note:

- For this question, you only need to modify CSI.cpp. The test codes are in main.cpp.
- We provide the basic test cases for the Autolab, and we will use a richer set of test cases for evaluation. We strongly encourage you to add more diverse test cases to make sure your application works as expected.

Background on CSI. CSI describes how Wi-Fi signals travel through space. We can determine which activities are performed by a person inside a room by analyzing the CSI values. A series of CSI values are recorded when two Wi-Fi devices in a room communicate with each other. In particular, when a data packet (a unit of data) is transferred, information is sent through multiple data transmission *channels* where a channel consists of multiple *subcarriers*. Here, a CSI value (represented as a complex number) is recorded per subcarrier to indicate the state of the corresponding subcarrier. Accordingly, for a single data packet, (*channels X subcarriers)* CSI values are recorded. In order to conduct useful analysis, we would like to convert these raw CSI values into “amplitude” values (meaning received strength of Wi-Fi signals). More details will come later.

Question 2-1: Printing Complex Values [1 Marks]

Objectives: Implement the following function in CSI.cpp to print the object of the custom struct `Complex` in a designated format.

- `std::ostream& operator<<(std::ostream &os, const Complex &c)`
    - This method overloads the insertion operator (`<<`) of this struct so that we can print out a Complex variable `z` using `std::cout << z`. The output should be: `a+bi` if the imaginary part is greater or equal to 0, and `a-bi` otherwise.
    - The purpose of this function is similar to overriding the `toString` method in Java!

Description: We will fill up a convenience struct called `Complex`, which represents a single raw CSI value. Remember, raw CSI values are expressed in complex numbers. As you are aware, a complex number `z = a + bi` has two parts: real and imaginary. The `Complex` struct stores the real and imaginary parts (a, b) as the integer type public member variables `real` and `imag`. All the other parts of the `Complex` struct are given.

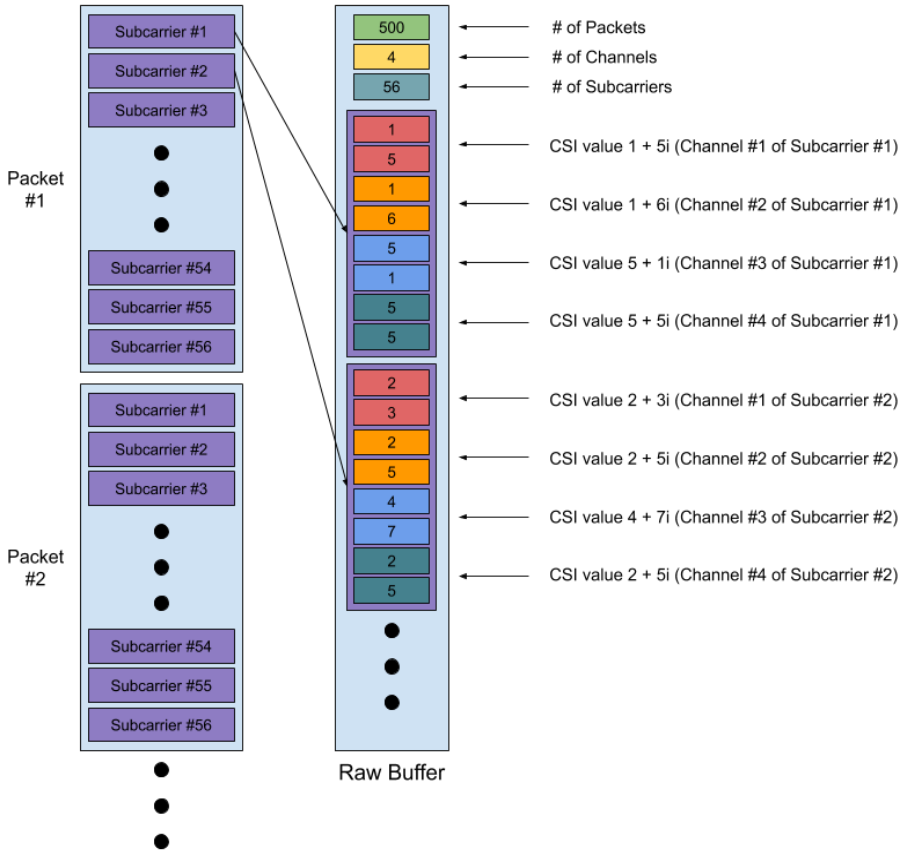
- -1e6 < real , imag < 1e6
- If imag >= 0, it should be represented as a+bi.
  ex1) 1+3i
  ex2) 1+0i
- If imag < 0, it should be represented as a-bi.
  ex1) 1-3i
  ex2) -1-3i

Question 2-2: Reading and Organizing CSI Values [1 Marks]

Objectives: Implement the following function in CSI.cpp to organize the input stream of CSI values in a file into a 2D array.

- void read_csi(const char* filename, CSI* csi)
  - It reads the input file (with the filename) containing raw CSI values and updates the csi object based on a description below.
  - Assume that the provided csi pointer is valid (i.e., the csi object has already been created before calling this function).

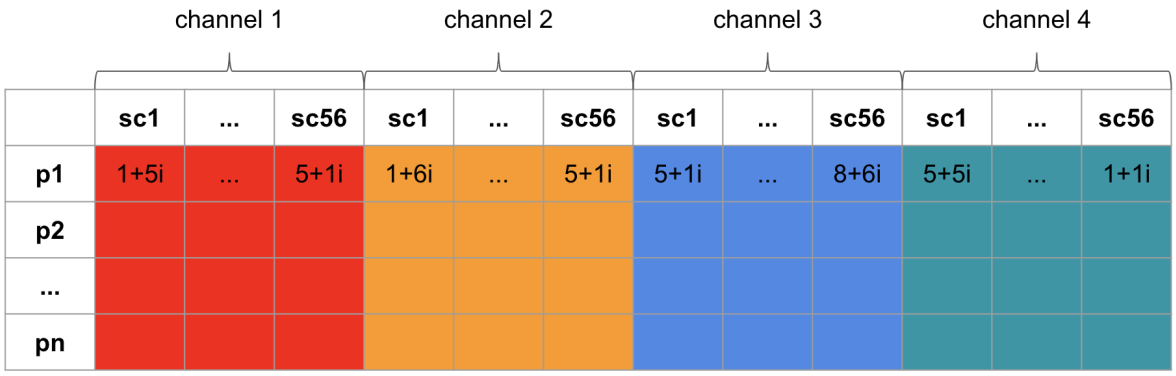
Description: The format of the input file is shown in the diagram below. All the values are integers.



In the example above, the first three lines of the input file indicate the number of packets delivered (i.e., 500) along with the number of channels (i.e., 4) and subcarriers (i.e., 56). Note that the number of data packets, number of channels, and number of subcarriers is not a fixed number, and you should read the information from the input file.

From the line 4 below, each line holds a raw CSI value. In this example, the file has 224 raw CSI values per packet since a packet is delivered through 224 subcarriers (56 * 4); in total, the file has 112,000 (500 * 56 * 4) raw CSI values. For each data packet, you can see that CSI values are grouped by subcarriers (not by channels).

For further analysis of CSI values, we will organize them into a 2D array such that each row represents a packet (see the figure below). In each row, the CSI values are grouped by channels starting from channel 1 and ending with channel 4. For each channel, the CSI values for 56 subcarriers are listed in order.



The `csi` object (passed as the parameter) has a member attribute, `data` to store a 2D array of `Complex` objects.

`csi->data[num_packets][num_channels * num_subcarriers]`.

You need to store the reorganized CSI data into this array. Also, update the properties such as `num_channels`, `num_subcarriers`, and `num_packets`. You must update these member variables; otherwise, the member functions of the CSI struct (such as `packet_length` and `print`) will not work as expected.

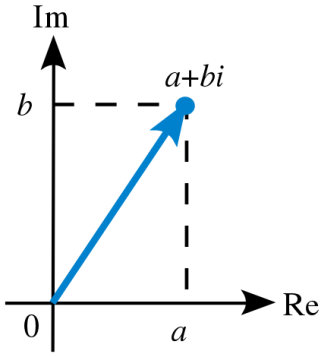
Hint: Use `std::stoi(std::string s)` to convert an integer string into an integer.

Question 2-3: Decoding CSI Values [1 Marks]

Objectives: Implement the following function in CSI.cpp to decode raw CSI values, i.e., converting the raw complex CSI values into amplitude values.

- `double** decode_csi(CSI* csi)`
  - It returns amplitude values that are converted from the provided csi object.
  - You should allocate the proper amount of memory space to store amplitude values.
  - Assume that the provided `csi` pointer is valid.

Description: The amplitude, amp, for a complex number, a+bi, can be computed as $amp = \sqrt{a^2 + b^2}$ using the Pythagorean Theorem.



Use the `sqrt()` function in `<cmath>`. The below code snippet shows how the amplitude can be calculated using the `sqrt()` function.

```
int a = 1, b = 2;
cout << sqrt( lcpp_x: a*a + b*b) << endl;
```

The resulting amplitude array should be in the same format as the data array in the input `csi` object. Specifically, the output array should have the same number of rows and columns as the `csi->data` array. Also, each element in the output array should contain the amplitude (in the `double` type) of the corresponding `Complex` object of the `csi->data` array.
Note that when comparing the values with the answer, we are going to compare up to three decimal places. If you use the `sqrt()` function in `<cmath>`, there will be no problem regarding the floating point precision.

Question 2-4: Extracting Useful Information [1 Marks]

Objectives: Implement the following function in CSI.cpp to calculate the median of CSI values per packet.

- `double* get_med(double** decoded_csi, int num_packets, int packet_length)`
    - It calculates a median of the decoded amplitudes per packet and returns medians for all packets in the 1D double array format.
    - You should allocate the proper amount of memory space (for `num_packets` double values) to store the values.
    - Assume that the provided `decoded_csi` pointer is valid.
    - DO NOT modify the original values in `decoded_csi`. We will reuse this variable in Question 2-5.

Description:

Calculate the median of the decoded amplitudes across all channels and subcarriers in the same packet inside the `get_med()` function. Do this for all packets and return the output in the form of the following 1D array (as shown below).

- If the `packet_length` is odd, the median is the middle value. For sorted elements a_1, a_2, …, a_n, median = $a_{(n+1)/2}$
- If the `packet_length` is even, the median is calculated as the mean of the two middle values. For sorted elements a_1, a_2, …, a_n, median = $(a_{n/2} + a_{n/2+1})/2$.

	p1	**p2**	**p3**	**...**	**pn**
median	10.779	3.535	5.323		0.254

Note that when comparing the values with the answer, we are going to compare up to three decimal places. Therefore, a small difference due to floating point precision issues will not be a problem.

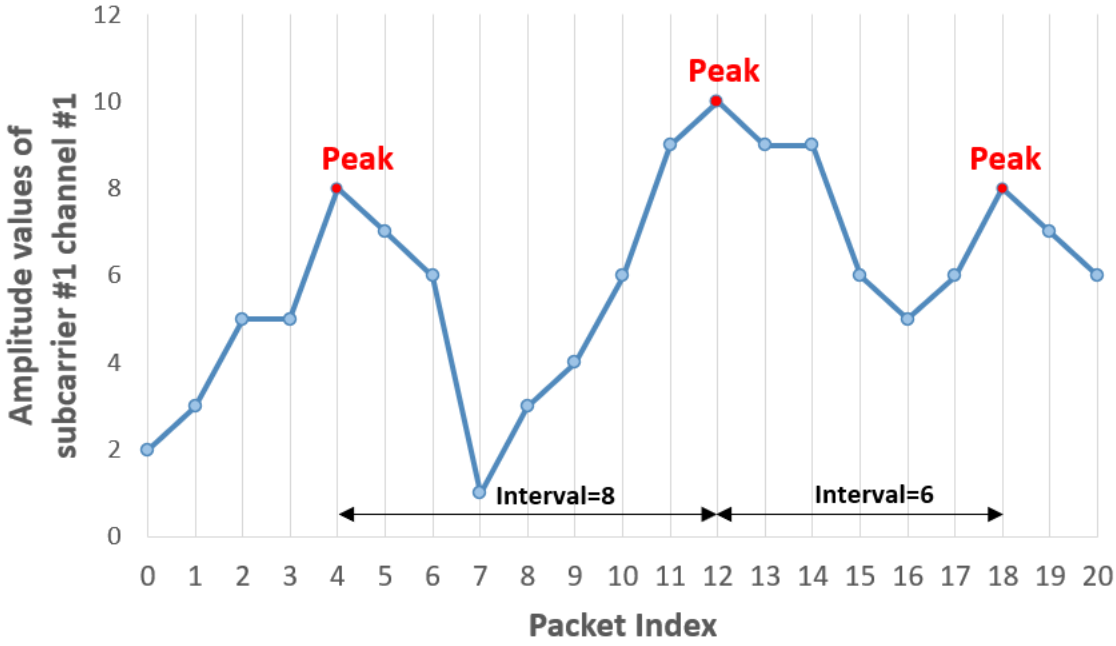
Question 2-5: Estimating Breathing Interval [1 Marks]

Objectives: Implement the following function in CSI.cpp to estimate the breathing interval from the CSI values.

- `double breathing_interval(double** decoded_csi, int num_packets)`
    - It returns the estimated breathing interval.
    - Assume that the provided `decoded_csi` pointer is valid.

Description: You discovered that the amplitude values of subcarrier #1 channel #1 (`decoded_csi[0][0]`, `decoded_csi[1][0]`, …, `decoded_csi[num_packets-1][0]`) are highly correlated

with a person's breathing pattern. Specifically, when the person inhales, the amplitude value of subcarrier #1 channel #1 increases and when the person exhales, the value decreases. Therefore, you can estimate the breathing interval by looking at the peak interval of the amplitude values. This procedure is illustrated in the below figure.



- For simplicity, let us assume that the peak value is defined as "the value which is strictly larger than the previous two values and the following two values". For example, in the above figure, the value at packet index 4 is a peak since it is larger than the value at packet index 2, 3, 5 and 6.
    - If some of the previous value / following values to compare doesn't exist, only compare with the existing ones. For example, for the value at packet index 0, you can just compare it with the value at packet index 1 and 2 since the packet index -1 and -2 doesn't exist. If it is larger than those two values, it becomes a peak.
    - When comparing the two values `a` and `b`, `b` is strictly larger than `a` if "`a + eps < b` (`eps = 1e-10`)".
- After the peak values are determined, calculate the average interval of the peak values, which is the estimated breathing interval. For example, in the above figure, the interval values are 8 and 6. Therefore you should return 7, which is the mean of the two values.
- If there is one or no peak value, return `num_packets`.