

Week 1- Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

Objective

Develop a C++ program leveraging modern C++ features to perform data processing tasks.

Background

In a meteorological monitoring system, we collect temperature data (in Celsius, °C) throughout the day. Due to possible sensor errors, we need to analyze this data and compute some statistical information. Your task is to write a C++ program to process these temperature readings.

Task

Given a `std::vector` containing temperature data (data type could be `int` or `double`), implement the following:

1. A **template function** `aveTemp` with **two** parameters to compute the **average temperature of high-temperature anomalies**, defined as the mean of all values greater than **35°C**.
2. An **overloaded template function** `aveTemp` with **three** parameters to compute the **average temperature of low-temperature anomalies**, defined as the mean of all values below **0°C and above -50°C**.
3. Correctly print these results.

Required C++ Features

- Template function
- Function overloading (distinguished by the number of parameters)
- Range-based for-loop
- `auto`
- Reference variables

Input: Please use the provided C++ file in the zip, ensuring that **the vectors defined in the file remains unchanged**.

Submit:

For a more efficient marking process, we kindly request that you submit each file individually, instead of in a zip file. Thank you for your cooperation.

1, **week1.cpp** : the c++ source code

2, **week1.txt**: a txt file contains the same c++ code as week1.cpp for Plagiarism Review.

(If you are unable to create the .txt file correctly, please configure your system to display file extensions.)

3, **output.jpg (or png, bmp)**: a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Week 2- Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

Task

In data processing, it is often necessary to merge two sorted arrays while keeping the merged array sorted. This assignment requires you to use **vector** and **iterator** to design a program that merges two sorted arrays while maintaining the order.

1. Input:

Two predefined sorted **vector**<int> objects in the code (**sorted in ascending order**).

Example 1: **the first vector**: `vector<int> vec1 = {1, 3, 3, 5, 7};` and

the second vector: `vector<int> vec2 = {2, 4, 6, 8, 9, 10, 11};`

Example 2: **the first vector**: `vector<int> vec_1 = {2, 4, 6, 8, 9, 10, 11};` and

the second vector: `vector<int> vec_2 = {1, 3, 3, 5, 7};`

Make sure your program works correctly with these two examples.

2. Merge Two Sorted Arrays:

< Do not use STL algorithms such as merge, as we will be learning them next week. Please implement this functionality yourself. >

Use **iterators** to traverse the vector objects and merge the two sorted arrays while **ensuring the merged array remains sorted (in ascending order)**.

3. Output:

Display the original two sorted arrays.

Display the merged sorted array.

Input: Please use the provided C++ file in the zip, ensuring that **the vectors defined in the file remains unchanged**.

Submit:

For a more efficient marking process, we kindly request that you submit each file individually, instead of in a zip file. Thank you for your cooperation.

1, **WA2.cpp** : the c++ source code

2, **WA2.txt**: a txt file contains the same c++ code as week2.cpp for Plagiarism Review.

(If you are unable to create the .txt file correctly, please configure your system to display file extensions.)

3, **output.jpg (or png, bmp)**: a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Week 3 - Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

Task Description:

You are given a vector of integers representing the scores of students in a test. Your task is to perform the following operations on the vector:

Task1: Use **count_if** and a **custom function** to count the number of scores that are greater than 85, and print this number.

Task2: Use **binary_search** to correctly check if the score -1 exists in the vector, and print the search result. (Tips: **binary_search** only works with sorted numbers).

Task3: If score -1 exists, use **remove** and **erase** to remove all occurrences of -1 from the vector, and then print the average of the scores.

Task4: Use **for_each** and a **custom function** to modify all the scores in the vector according to the rules below, and print the average of the updated scores.

- change score ≥ 85 into 7
- change $75 \leq \text{score} < 85$ into 6
- change $65 \leq \text{score} < 75$ into 5
- change $50 \leq \text{score} < 65$ into 4
- change score < 50 into 3

Input:

Please use the provided WA3.cpp file and do not modify the given vector.

Output:

```
Task1: Number of scores greater than 85: 5
Task2: Score -1 exists
Task3: Average of scores after removing -1: 58.2333
Task4: Average of updated scores: 4.76667
```

Submit:

- 1, **WA3.cpp** : the c++ source code
- 2, **WA3.txt**: a txt file contains the same c++ code as WA3.cpp for Plagiarism Review.
(If you are unable to create the .txt file correctly, please configure your system to display file extensions.)
- 3, **output.jpg (or png, bmp)**: a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Week 4- Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

This assignment is to design and implement a C++ class named `TimeDuration` that represents a time interval in hours, minutes, and seconds. The class should support arithmetic operations (addition and subtraction), comparisons, formatted output, and internal normalization of time values.

Your implementation must demonstrate the following C++ features:

1. Member Variables
At least three **private** member variables: hours, minutes, seconds
2. Constructors
A **default constructor** that initializes all values to -1.
3. Destructor
A **destructor** that prints a message displaying **the time duration** in the format, for example, `TimeDuration(1h 30m 30s)`
4. Member Functions
`int getHours() const, getMinutes() const, getSeconds() const,`
`void setHours(int), setMinutes(int), setSeconds(int),`
`int totalSeconds() const` — returns the total duration in seconds.
5. Must implement the following operators and declare them as **const** member functions
`==` : Returns true if two durations have the same total seconds.
`<, >` : Return true or false by comparing two durations based on total seconds.
`+, -` : Return total seconds by Add or Subtract two `TimeDuration` objects.
`<<` : Output the duration in the format, for example, `TimeDuration(2h 36m 15s)`

Besides, you need to organize your program using **three separate files**:

- `TimeDuration.hpp` — class declaration (header file)
- `TimeDuration.cpp` — function implementation file
- `run_time.cpp` — main program to test the class (provided in WA4.zip)

Input:

Please use the provided `run_time.cpp` to test your implementation.

Don't modify the main function in `run_time.cpp`

Output: (should be like the below)

```
Output1 >> t1 : TimeDuration(-1h -1m -1s)
          >> t1 : TimeDuration(1h 30m 30s)
Output2 >> t1 = t2 : false
          t1 > t2 : true
          t1 < t2 : false
Output3 >> t1 + t2 : 7200 s
          t1 - t2 : 3660 s
Output4 >>
Destroying: TimeDuration(0h 29m 30s)
Destroying: TimeDuration(1h 30m 30s)
```

Submit:


- 1, **all C++ source code**: TimeDuration.hpp, TimeDuration.cpp, run_time.cpp
- 2, **WA4.txt**: a txt file contains all the source code.
- 3, **output.jpg** (or png, bmp): a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Vscode: How to compile and run all files in a folder when there is only one main()

- 1 Select the folder that contains the C/C++ files you want to compile.

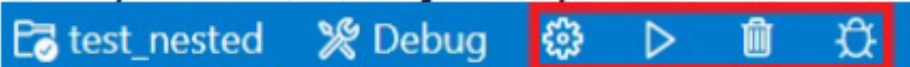
You can select the folder by the quick pick menu from the status bar.







- 2 Optional: Select either debug or release mode for building the binary (debug is the default case).



- 3 Now you can build/run/debug the binary.



-  Build: This task will compile all C/C++ files in the selected folder and will link them into a binary.
-  Run*: This task will execute the built binary.
-  Clean*: This helper task will delete all files in the build dir.
-  Debug*: This task will start a debugging session for the binary.

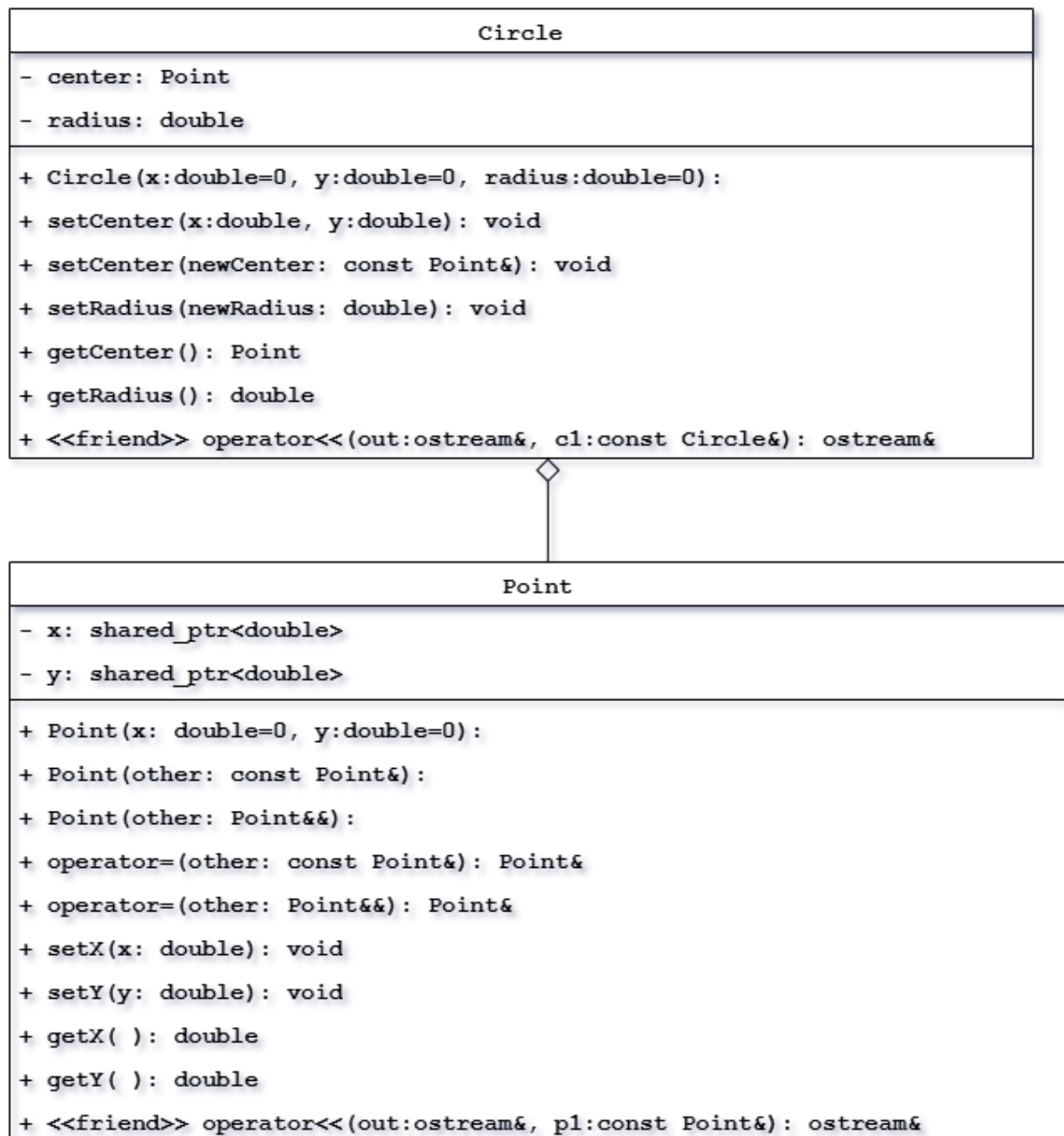
Week 5- Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

Objective: This assignment aims to deepen your understanding of advanced C++ class features, including **constructors**, **assignment operators**, **move semantics**, **friend functions**, and **class aggregation**. You will implement two related classes—Circle and Point—to explore their interactions and gain hands-on experience with core object-oriented programming principles in C++.

Description:

- In this assignment, you will implement two C++ classes, Circle and Point, based on the provided UML diagram. And use **the given main function** to produce the output shown below.
- The Point class represents a point in 2D space using x and y coordinates.
- The Circle class models a circle defined by a Point object as its center and a double representing its radius.
- The friend function operator<< in the Point class should print the point's details. If the Point object has been moved and is empty, it should output ***null*** instead of dereferencing an invalid pointer.



Input:

Please use the provided **run_wa5.cpp** to test your implementation.

Don't modify the **main** function in the cpp.

Output: (should be like the below)

```

=====
Output 1:
p1 Point(1, 1)
p2 Point(2, 2)
-----
p1 Point(1, 1)
p2 Point(3, 3)
=====

Output 2:
p2 Point(null,null)
p3 Point(3, 3)
-----
p1 Point(null,null)
p3 Point(1, 1)
=====

Output 3:
p3 Point(1, 1)
c1 Circle(center: Point(10, 10), radius: 0)
-----
c1 Circle(center: Point(10, 10), radius: 0)
c2 Circle(center: Point(20, 20), radius: 0)
=====

Output 4:
c2 Circle(center: Point(20, 20), radius: 0)
c3 Circle(center: Point(30, 30), radius: 0)
-----
c2 Circle(center: Point(null,null), radius: 0)
c3 Circle(center: Point(20, 20), radius: 0)
=====

```

Submit:

1, all C++ source code:

Organizing the source code into separate files **is not mandatory**.

You can consolidate all code into a single cpp file.

2, **WA5.txt**: a txt file contains all the source code.

3, **output.jpg (or png, bmp)**: a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Week 6- Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

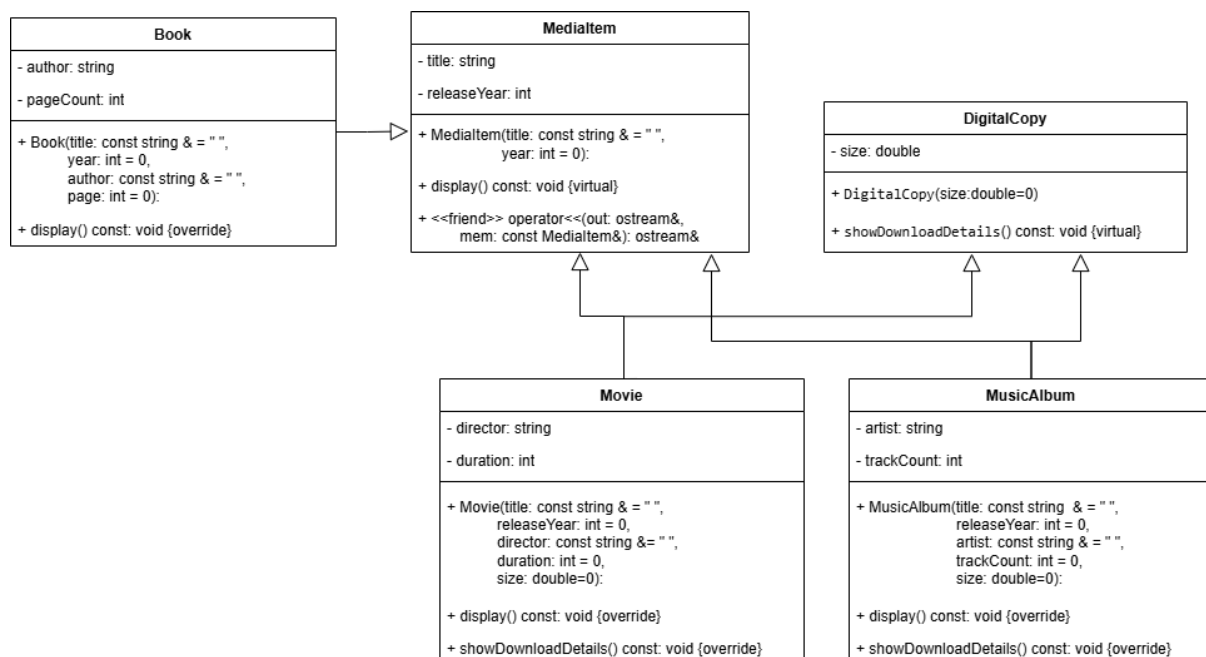
In this assignment, you will design and implement a set of C++ classes that model a simple media library system.

You are required to implement the classes strictly based on the provided UML diagram.

The system includes books, movies, and music albums, with support for digital download information, and demonstrates concepts of inheritance, polymorphism, and virtual functions.

You must:

- Analyze the UML and the provided main() function carefully. **Ensure** that your class design is **compatible with the provided main() function** and can **produce the expected output**.
- Appropriately use inheritance, virtual functions, override specifiers, and friend functions.
- Correctly define and implement all classes according to the UML diagram.
 - Only implement the member variables and functions explicitly listed in the UML.
 - Do not add helper functions, extra attributes, or other changes beyond the UML design.
 - Do not omit any required function.



Input:

Please use the provided ***run_wa6.cpp*** to test your implementation.

Don't modify the **main** function in the cpp.

Output: (should be like the below)

```
Output 1: Title: CODE, Year: 2000, Author: Charles Petzold, Pages: 400
Output 2: Title: Inception, Year: 2010, Size: 1000 MB, Director: Christopher Nolan, Duration: 148 min
Output 3: Title: Thriller, Year: 1982, Size: 20 MB, Artist: Michael Jackson, Tracks: 9
```

Submit:**1, all C++ source code:**

Organizing the source code into separate files **is not mandatory**.

You can consolidate all code into a single cpp file.

2, WA6.txt: a txt file contains all the source code.**3, output.jpg (or png, bmp):** a screenshot of the output by your program

Please refer to the submission page for the Marking Rubric.

Week Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

The goal of this assignment is to develop a Student Grade Management System using modern C++ programming techniques. The system should allow you to add students, record and validate their grades, calculate average scores, identify top and failing students.

Tasks:

1. Define a NoGradesException Class

- Inherit from `std::exception`.
- Store an error message as a private `std::string`.
- Construct the message like:
`" Bob: No grades available."`
- Override the `what()` method.

2. Implement a Student Class

This class represents an individual student.

- **Private members:**
 - `std::string name`
 - `std::vector<double> grades`
- **Constructor:**
 - Accepts the student's name.
- **Public methods:**
 - `void addGrade(double grade)`
 - Adds a grade to the student.
 - Throws `std::invalid_argument` if the grade is not between 0.0 and 100.0.
 - `double averageGrade() const`
 - Returns the average of the grades.
 - Throws a **NoGradesException** if the student has no grades.
 - `std::string getName() const`
 - Returns the name of the student.
 - `bool hasGrades() const`
 - checks whether the student has any grades recorded.

3. Implement a StudentManager Class

This class manages a group of Student objects.

- **Private member:**
 - `std::vector<Student> students`
- **Public methods:**
 - `void addStudent(const Student&)`
 - Adds a student to the system.
 - `void filterFailingStudents() const`
 - Prints failing students whose average is **below 50.0**.

- Skips students with no grades.
- Prints “None.” if no failing students.
- void printTopStudents(int n) const
 - Print the top n students based on the average grade, along with their average grade, formatted to **two decimal places**, like
Kevin: 76.66
 - Prints “None.” if no students.
- void operator()() const
 - Prints students with their average grade, formatted to two decimal places.
 - Prints “None.” if no students.

Input:

Please use the provided **run_wa7.cpp** to test your implementation.

Don't modify the main function in **run_wa7.cpp**

Output: (should be like the below)

```
Output 1:
Invalid grade. Must be between 0.0 and 100.0.
Bob: No grades available.
```

```
Output 2:
Student Management System:
None.
-----
Failing students:
None.
-----
Top 2 students:
None.
```

```
Output 3:
Student Management System:
Alice: 83.58
Bob: No grades available.
Charlie: 42.50
Kevin: 55.00
-----
Failing students:
- Charlie
-----
Top 2 students:
1. Alice : 83.58
2. Kevin : 55.00
```

Submit:

For a more efficient marking process, we kindly request that you submit each file individually, instead of in a zip file.

1, all C++ source code

2, **wa.txt**: a txt file contains all the source code for plagiarism review.

(If you are unable to create the .txt file correctly, please configure your system to display file extensions.)

3, **output.jpg** (or png, bmp): a screenshot of the output by your program.

Please refer to the submission page for the Marking Rubric.

Week 8 - Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

Objective:

- Implement a custom, robust matrix operation program.
- Apply multithreading (`std::thread`) to enhance computational efficiency for large matrix operation.
- Compare the performance of single-threaded and multithreaded computations.

Tasks:

A custom matrix operation is defined as follows:

Given two matrices, where A is an $m \times n$ matrix and B is an $n \times p$ matrix, their result C is an $m \times p$ matrix, with each element $C[i][j]$ computed according to the following steps:

1. Compute the dot product of the i -th row of A and the j -th column of B , defined as:

$$\text{dot_product} = \sum_{k=1}^n A[i][k] \times B[k][j]$$

2. If the remainder of the dot product divided by 100 is greater than 50, set $C[i][j] = 1$.
3. Otherwise, set $C[i][j] = 0$.

Then,

1. Implement the matrix operation using a single-threaded approach.

`customOpSingle`

2. Implement the matrix operation using a multithreaded approach.

`customOpMulti`

Input:

Please use **the provided `run_wa8.cpp`** to test your implementation.

Don't modify the provided code!

Submit:

- 1, **all C++ source code**
- 2, **wa8.txt**: a txt file contains all the source code for *plagiarism review*.
- 3, **output.jpg** (or png, bmp): a screenshot of the output by your program.

Please refer to the submission page for the Marking Rubric.

Week 9 - Assignment (3 marks)

Note: Try to make the best use of appropriate C++ features.

You are provided with a working implementation of a template-based SkipList<Key, Value> class. This data structure supports fast search, insertion, and deletion by maintaining multiple forward pointers at different levels for each node.

Tasks:

Your task is to extend the SkipList class by implementing a **range query** operation. Ensure your range query method performs efficiently by leveraging the skip list structure. This operation should:

- Retrieve all key-value pairs where the key falls within a specified range low, high (inclusive).
- Collect the results into a `std::vector<std::pair<K, V>>`.
- Return the vector to the caller.

Input:

Please use ***the provided run_wa9.cpp*** to test your implementation.
Don't modify the provided code!

Submit:

- 1, **all C++ source code**
- 2, **wa9.txt**: a txt file contains all the source code for *plagiarism review*.
- 3, **output.jpg** (or png, bmp): a screenshot of the output by your program.

Please refer to the submission page for the Marking Rubric.

Week 10 - Assignment (3 marks)

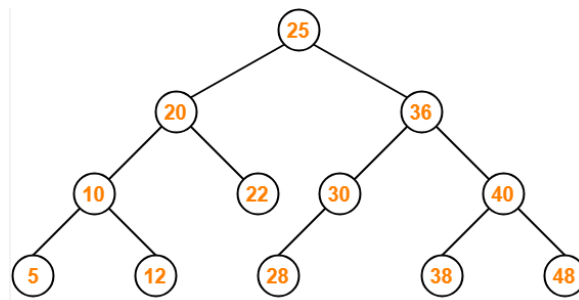
Note: Try to make the best use of appropriate C++ features.

Extend the provided Treap class with additional functionality to deepen your understanding of Treap operations and the use of augmented data structures in binary search trees.

Tasks:

You are provided with a basic implementation of a Treap class that supports fundamental operations such as insertion, deletion, and search. Your task is to extend this implementation by augmenting each node with additional information and modifying relevant functions accordingly.

1. **Subtree Key Sum:** Modify each node to maintain the **sum** of all keys in its subtree, **including itself**. For example, if node 20 has subtree nodes 10, 22, 5, and 12, then the sum stored at node 20 should be 69 ($= 20 + 10 + 22 + 5 + 12$).



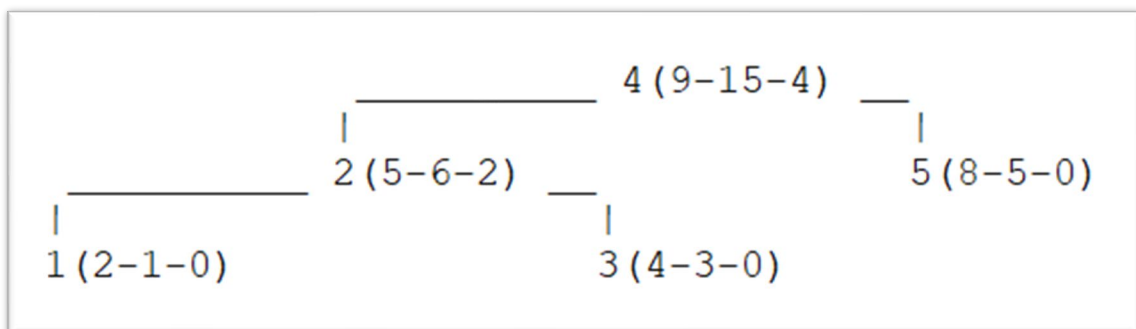
2. **Subtree Node Count (Excluding Self):** Add a field to each node to track the **count** of nodes in its subtree, **excluding the node itself**.
3. **Modified Output Function:** Update the print-related functions to produce the required **output** below. For each node, it should display the format: **key (priority-sum-count)**

Input:

Please use *the provided run_wa10.cpp* to complete these tasks.

Don't modify the main function!!!

Output: (should be like the below) **key (priority-sum-count)**



Submit:

- 1, **all C++ source code**
- 2, **wa10.txt**: a txt file contains all the source code for *plagiarism review*.
- 3, **output.jpg** (or png, bmp): a screenshot of the output by your program.

Please refer to the submission page for the Marking Rubric.