2800ICT In-Trimester Test – 2024

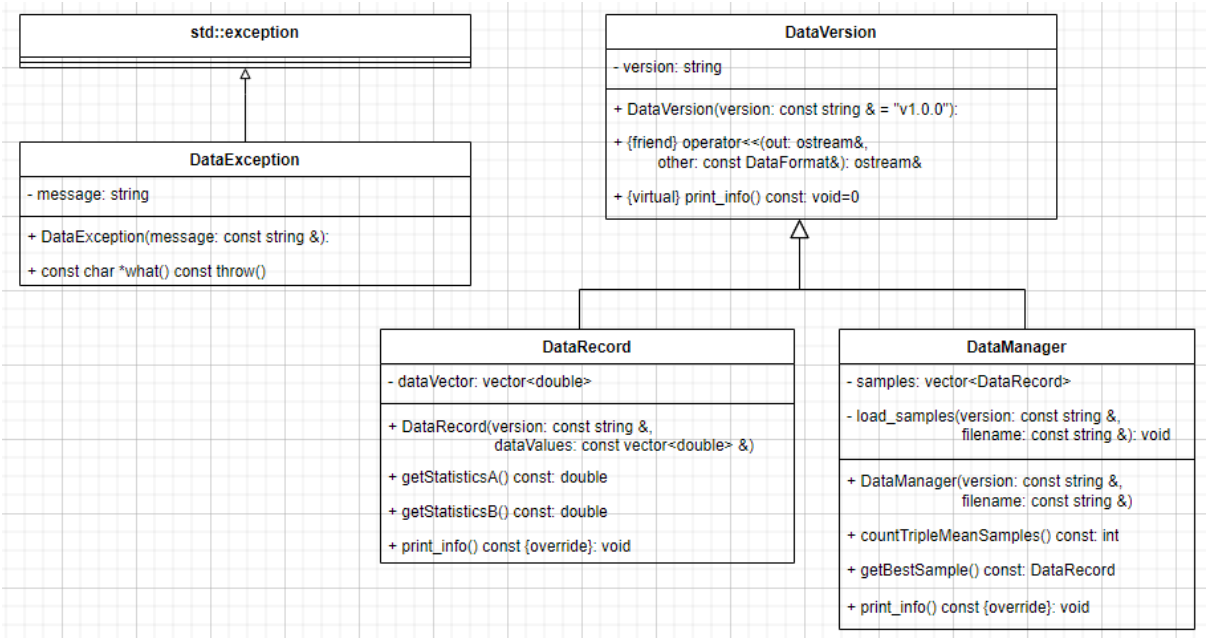
Question 1 (10 Marks)

You must maximise your use of modern C++.

Description:

Based on the supplied q1.cpp, you're tasked with building a simple data analysis system called **DataManager** to manage and analyse a collection of **DataRecord**s. The **DataRecord** holds a series of numerical values. The **DataManager** should be capable of performing collective analysis across all samples. Additionally, you are required to implement a custom exception class **DataException**; and **DataRecord**'s constructor should throw an error when its data contains negative values. It's crucial to note that you must not alter anything in the *main()* function in the given q1.cpp.

The class UMLs are like below:



where

- **`DataManager::countTripleMeanSamples`**: This function returns the number of samples where the maximum value exceeds three times their mean value.
- **`DataManager::load_samples`**: This function is responsible for loading data from a file.
- **`DataManager::getBestSample`**: This function returns the sample with the highest mean value.
- **`DataRecord::getStatisticsA`** and **`DataRecord::getStatisticsB`** need to be implemented to calculate specific statistics values. These functions will be utilized by other functions to achieve the desired **Output**.
- **`DataManager::print_info`** and **`DataRecord::print_info`** should be adapted to print the desired **Output**.

Importantly, you're required to add two appropriate member functions to the above **DataManager** class to ensure the main function runs smoothly and produces the desired output.

Input:

```
//================================================
// MUST NOT CHANGE ANYTHING IN THE MAIN FUNCTION
//================================================
int main()
{
    DataManager manager;
    try
    {
        manager("v1.5.0", "wrong_input.txt");
    }
    catch (const exception &e)
    {
        cout << e.what() << endl<<endl;
        manager("v1.5.0", "input_datarecords.txt");
        manager.print_info();
    }
    return 0;
}
```

Output:
For the supplied input_datarecords.txt, it should output like below.

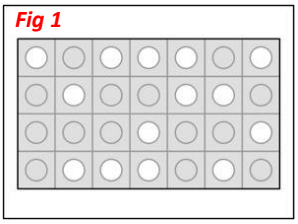
```
Cannot open file: wrong_input.txt

Error: Negative values are not allowed in the data sample.
Error: Negative values are not allowed in the data sample.
Count of TripleMeanSamples: 6
Details of the sample with the max Mean:
Data Version: v1.5.0 | Max Value: 954.074 | Mean Value: 91.0583
```

Submit:
1, All C++ source code: *.cpp and *.hpp if your code is organized into separate files.
Organizing the source code into separate files is not mandatory.
You can consolidate all code into a single cpp file.
2, q1.txt: a txt file contains all the source code.
3, q1.jpg (q1.png or q1.bmp): a screenshot of the output by your program.

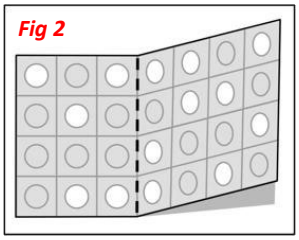
Question 2 (10 Marks)

Develop a C++ program to solve this question. The usage of class is not mandatory.

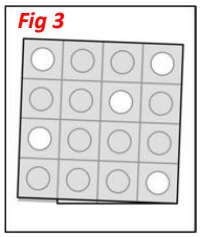
Description:



During the early days of computer programming, instructions were encoded using designated cards. Imagine a card arranged in a grid pattern with M rows and N columns (Fig. 1). Each cell within this grid could potentially contain a small, removable paper dot. If the dot is removed, it leaves behind a hole, thus each cell is marked by the presence or absence of such a hole.



You are tasked with configuring a card device, but the challenge is that it's not equipped to create new holes. Fortunately, there's already a card at your disposal. The strategy involves folding the card vertically in such a place that it results in the rightmost column overlaying to the left. This fold aligns certain cells on the grid. Take, for instance, a card with 7 columns (Fig. 1). By folding it between the 3rd and 4th columns (Fig. 2), the adjacent cells from columns 3 and 4, 2 and 5, as well as 1 and 6, will align respectively. The 7th column will then dangle, not aligning with the initial card layout. The newly reconfigured card now consists of 4 columns (Fig. 3), with the order from left to right being the dangling 7th column, then the pairs (1,6), (2,5), and finally (3,4). Holes will appear in this altered configuration only where both corresponding cells had holes in the pre-folded card.



If you fold a card along a specified column, how will the appearance of the card change post-folding?

Input: Your program must be capable of receiving inputs from the user/keyboard. The first line of input contains three integers M ($1 \le M \le 10^3$), which is the number of rows, N ($2 \le N \le 10^3$ and $M \times N \le 10^3$), which is the number of columns, and K ($1 \le K \le N - 1$), which indicates that you plan to fold the card between column K and column K + 1. The next M lines describe the card. Each of these lines contains a string of length N containing only the characters '#' and 'o'. An '#' represents a cell without a hole and an 'o' represents a cell with a hole.

Output: Display the folded card.

Sample input and output are like below:

Input 1: 3 3 1 o## #o# ##o Output 1: ## ## o#	Input 2: 3 3 2 o## #o# ##o Output 2: o# ## ##	Input 3: 2 3 1 oo# #oo Output 3: #o o#	Input 4: 4 7 3 o#ooo#o #o##oo# ###o##o #ooo#o# Output 4: o##o ##o# o### ###o

Submit:

1, All C++ source code: *.cpp and *.hpp if your code is organized into separate files.
Organizing the source code into separate files is not mandatory.
You can consolidate all code into a single cpp file.

2, q2.txt: a txt file contains all the source code.

3, q2.jpg (q2.png or q2.bmp): a screenshot of **the above four sample inputs and outputs** by your program.

Please refer to the submission page for the Marking Rubric.