Assignment 2 (10%)

Due Date: 3 April 2024, 23:59 PM

1. This assignment contains **4** questions. You are required to finish a complete C++ program (.cpp only) for each question on E-quiz before the deadline.

2. You can check as many times as you want before the deadline. We will grade your **highest version**.

3. Only a small set of the test cases are visible for the testing, and a more comprehensive set of test cases will be used for grading. In other word, passing all the visible test cases does not mean that you can get the full mark for this question. Try your best to thoroughly check your program.

4. Hidden test cases will not be released.

5. The marking of each question is based on the percentage of the total test cases that your solution can pass. If your submitted solution leads to a compilation error on E-quiz, zero mark will be given to that question and no manual checking to your solution is provided in such a case.

6. No late submission will be accepted.

7. Plagiarism check will be performed.

8. You only need to use the material from Lectures 1 to 9. It is **NOT** necessary to include any other library, except <iostream> <fstream><string>.

9. You need to check your solution and submit it on E-quiz.

To ensure timely feedback to the students' questions, each of the following TAs will be responsible for one question. If you have any questions, you can contact the corresponding TA directly.

Question	TA responsible	Email Address
Q1	SANIYAZOV Dias	dsaniyazo2-c@my.cityu.edu.hk
Q2	Muratov Ali	amuratov4-c@my.cityu.edu.hk
Q3	MUKASHEV Alikhan	amukashev2-c@my.cityu.edu.hk
Q4	Hong Chenhao	HONG.Chenhao@my.cityu.edu.hk

Q1: [Maximum Subsequence]

In this question, you need to write a program to get the **maximum** subsequence from an input string. The program accepts two inputs, where one is an input string **I** (e.g., "abc", "123", "Abc123") and another one is a positive integer **K**, indicating the required length of the subsequence.
A subsequence contains a subset of the input string arranged in the original order. Please note that a subsequence must be continuous (e.g., "**123**" is a subsequence of "**123456**", but "**135**" is not a subsequence). You can use ***strcmp*** to compare two strings.

Note 1: The input string I consists of at most 50 characters.
Note 2: **K** is larger or equal to 1 and smaller or equal to the number of characters in **I**.
Note 3: No validity check is required for the inputs.

Example 1:

```
Please input a string:
1265432
Please input the length of subsequence:
2
The maximum subsequence is:65
```

Example 2:

```
Please input a string:
AbCd687fs
Please input the length of subsequence:
4
The maximum subsequence is:d687
```

Q2: [Sum of Dices]

Write a program that simulates a game of rolling **1~3** dices. The program should allow the user to enter the number of dice they want to roll, then generate and display all possible combinations of dice rolls, along with the number of occurrences for each sum of points.
A dice has 6 faces, each with one unique value: 1, 2, 3, 4, 5, and 6. The face value of a dice means the value showing upward.

The code framework is given. What you need to do is following:

1. Implement the `findSum1, findSum2, findSum3` of the Combination class to correctly calculate and store the combinations of dice rolls based on the number of dice provided.
2. Implement the `sortBySum` member function to sort the combinations in descending order based on the sum of points.
3. Implement the `sortByOcc` member function to sort the combinations in descending order based on the number of occurrences.
4. You are free to add a member function to the combination class to better fulfil the requirements.
5. Test the program with different numbers of dice and ensure that the combinations and occurrences are calculated and displayed correctly. The number of dices is fixed to **1~3**.

Additional question:
Like above, simulates a game of rolling **1~20** dices and generate and display all possible combinations of dice rolls, along with the number of occurrences for each sum of points. (Hint: You may need to use **recursion**.)

Note: No validity check is required for the inputs.

Code Framework

```
#include <iostream>
using namespace std;

class Combination {
   private:
        int dNum;// number of dices
       int cNum;// number of different point sums
       long long int **pPtr; // pointing to a 2D array for each pair
of occurrence and sum. Use long int to avoid overflow.
   public:
       Combination(int i = 1) {
/************** to be finished **************/
   }
       ~Combination() {
/************** to be finished **************/
/***********Remember to release memory*******/

       cout << "Memory is released" << endl;
       }

       void findSum1();
       void findSum2();
       void findSum3();
```

```
/************** to be finished **************/

       void sortBySum();
       void sortByOcc();
};


void Combination::findSum1() {

}

void Combination::findSum2() {

}
void Combination::findSum3() {

}


void Combination::sortBySum() {

}

void Combination::sortByOcc() {

}

void display(Combination &com) {

   int n;
   do {
       cout << "~~~~~~~~~~~~~~~~~" << endl;
       cout << "0 exit" << endl;
       cout << "1 sort by sum" << endl;
       cout << "2 sort by occurence" << endl;
       cout << "~~~~~~~~~~~~~~~~~" << endl;

       cin >> n;

       switch(n) {
           case 0: cout << "Bye!" << endl; break;
           case 1: com.sortBySum(); break;
           case 2: com.sortByOcc(); break;
       }
       cout << endl;

   } while(n != 0);

}

int main() {

   int diceNum;
   cout << "Enter the number of dice:" << endl;
   cin >> diceNum;
```

```
   Combination com(/****/);

   display(com);

   return 0;
}
```

Examples:

Example 1

```
Enter the number of dice:
2
~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~
1
1 occurrence(s) of sum 12
2 occurrence(s) of sum 11
3 occurrence(s) of sum 10
4 occurrence(s) of sum 9
5 occurrence(s) of sum 8
6 occurrence(s) of sum 7
5 occurrence(s) of sum 6
4 occurrence(s) of sum 5
3 occurrence(s) of sum 4
2 occurrence(s) of sum 3
1 occurrence(s) of sum 2

~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~
2
6 occurrence(s) of sum 7
5 occurrence(s) of sum 8
5 occurrence(s) of sum 6
4 occurrence(s) of sum 9
4 occurrence(s) of sum 5
3 occurrence(s) of sum 10
3 occurrence(s) of sum 4
2 occurrence(s) of sum 11
2 occurrence(s) of sum 3
1 occurrence(s) of sum 12
1 occurrence(s) of sum 2
```

```
~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~
0
Bye!

Memory is released
```

Example 2

```
Enter the number of dice:
4
~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~
1
1 occurrence(s) of sum 24
4 occurrence(s) of sum 23
10 occurrence(s) of sum 22
20 occurrence(s) of sum 21
35 occurrence(s) of sum 20
56 occurrence(s) of sum 19
80 occurrence(s) of sum 18
104 occurrence(s) of sum 17
125 occurrence(s) of sum 16
140 occurrence(s) of sum 15
146 occurrence(s) of sum 14
140 occurrence(s) of sum 13
125 occurrence(s) of sum 12
104 occurrence(s) of sum 11
80 occurrence(s) of sum 10
56 occurrence(s) of sum 9
35 occurrence(s) of sum 8
20 occurrence(s) of sum 7
10 occurrence(s) of sum 6
4 occurrence(s) of sum 5
1 occurrence(s) of sum 4

~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~
2
146 occurrence(s) of sum 14
140 occurrence(s) of sum 15
140 occurrence(s) of sum 13
125 occurrence(s) of sum 16
125 occurrence(s) of sum 12
```

```
104 occurrence(s) of sum 17
104 occurrence(s) of sum 11
80 occurrence(s) of sum 18
80 occurrence(s) of sum 10
56 occurrence(s) of sum 19
56 occurrence(s) of sum 9
35 occurrence(s) of sum 20
35 occurrence(s) of sum 8
20 occurrence(s) of sum 21
20 occurrence(s) of sum 7
10 occurrence(s) of sum 22
10 occurrence(s) of sum 6
4 occurrence(s) of sum 23
4 occurrence(s) of sum 5
1 occurrence(s) of sum 24
1 occurrence(s) of sum 4

~~~~~~~~~~~~~~~~~~
0 exit
1 sort by sum
2 sort by occurence
~~~~~~~~~~~~~~~~~~
0
Bye!

Memory is released
```

Q3: [Document Correction and Alignment Problem]

Write a program to remove the word duplication and format text in an input file.

Requirements:

1. **File input**: **Load the original text** from **input txt files** and **output the adjusted text** to **cout**. Assume that the whole input text contains at most 1000 words. Input txt files are in the same folder with this cpp program. The format of input filename is from "input1.txt" to "input10.txt".
2. **Remove duplication**: If there is a consecutive sequence of the same word, such as "*the the the world*", then correct it by deleting the duplication and leaving the single word. The corrected text should be "*the world*".
   1) All punctuations are treated as a part of a word. Therefore, "*world*" and "*world*," are not duplicate.
   2) Uppercase and lowercase letters are the same, for example, "*The the sky*" needs to be corrected to "*The sky*".
3. **Alignment**: The aligned text should have exactly 70 characters per line, including spaces and punctuations, with the exception of the last line of each paragraph. 1) Spaces should be distributed between words as evenly as possible. E.g., if one line can contain 4 words and 5 spaces, then the amounts of spaces between two consecutive words should be 2, 2 and 1.
4. **Numbering format**: If there is a digit (0-9) followed by a full stop (.), we take them as a numbering. Adjust the format so that each numbering is placed at the very beginning of each paragraph and followed by **one** space (see examples).

input1.txt:

Write a a program to compute compute the smallest smallest number of steps
needed to capture the target.
Requirements: 1. Input the the original matrix. Inputs includes: the size of the matrix, the position of the Cannon, the position of the target and the positions positions of the obstacles. You can can can assume that there are only one Cannon and one target, but but the number of obstacles are not clear. 2. Output the smallest number of steps needed to capture the target. If there is no solution, print a message: "No solution". 3. The program should detect the inputs. When the inputs inputs inputs are invalid, output error message: "Invalid inputs" and ask the user to input again.

input2.txt:

Declare your own functions as following: 1. char *mystrncat (char *destination, const const char *source, int num); Appends the first num characters of source to destination. If the the length of the string in source is less than num, only only the content up to the terminating nullcharacter is copied. Assume assume that the destination is large large enough to contain the concatenated resulting string. 2. char *mystrtok (char *str, const char *delimiters delimiters); A sequence of calls to this function split str into into tokens, which which are sequences of contiguous characters separated by any of the characters that are part of delimiters. For more details about this function, you can refer website.

Example:

Example 1:

```
Please input file number (1-10):
1
Write a program to compute the smallest number of steps needed to
capture the target.
Requirements:
1. Input the original matrix. Inputs includes: the size of the matrix,
the position of the Cannon, the position of the target and the
positions of the obstacles. You can assume that there are only one
```

```
Cannon and one target, but the number of obstacles are not clear.
2. Output the smallest number of steps needed to capture the target.
If there is no solution, print a message: "No solution".
3. The program should detect the inputs. When the inputs are invalid,
output error message: "Invalid inputs" and ask the user to input
again.
```

Example 2:

```
Please input file number (1-10):
2
Declare your own functions as following:
1. char *mystrncat (char *destination, const char *source, int num);
Appends the first num characters of source to destination. If the
length of the string in source is less than num, only the content up
to the terminating null-character is copied. Assume that the
destination is large enough to contain the concatenated resulting
string.
2. char *mystrtok (char *str, const char *delimiters delimiters); A
sequence of calls to this function split str into tokens, which are
sequences of contiguous characters separated by any of the characters
that are part of delimiters. For more details about this function, you
can refer website.
```

Q4: [Library borrowing System]

In this question, you are required to design a simple library borrowing system and finish tasks as the followings:

1. Accept the information about the books, including book name, data, state and subject.
2. Assign the books into different subject categories according to the input.
   For simplicity, assume:
   - There are only **three** subject categories in this small library.
   - Each book only can be assigned into **one** subject.

3. List all the books and their information under each subject category respectively. The books listed in each subject should be sorted by date (newest to oldest).
4. Accept the name of the book that the reader want to borrow. If the book is unavailable, print a message and ask the reader to input again.

Requirements:

1. The concept of *Library* is implemented as a C++ class.
   - Each *Library* object owns three Subject objects, named Art, Science, and History.
     Hint: the use of an array to keep three Subject objects will make your program significantly more complex than keeping them as three separate variables.
2. The concept of *Subject* is implemented as a C++ class.
   - Each *Subject* contains its own list of *Books* and the count of *Books*.
     Hint: here you can create an object array to store all the *Books*.
3. The concept of *Book* is implemented as a C++ class.
   - Each *Book* object has its own Name (csting), Date (int) and State (bool, 1: available, 0: not available).
4. **All data members of each object can only be accessed by the object itself.**

More description about the above three classes

Ideas on the data members of the **Subject** class. [The class may contain other data members]		Ideas on the data members of the **Library** class. [The class may contain other data members]		Ideas on the data members of the **Book** class. [The class may contain other data members]	
count	The number of Books	Name	The name of this Library object	Name	The name of this book object
list	An array of Books	Art	The first Subject owned by the Library	Date	The date of this book object
		Science	The second Subject owned by the Library	State	The state of this book object
		History	The third Subject owned by the Library		

Hint: You can assume that: 1) the number of Books is at least 1 and at most 100; 2) any cstring consists of at most 10 characters; 3) all user inputs are valid;4) name and date of different book won't be conflicted;5) When there is no books in a subject, we only print the subject name.

Example:

```
Input the number of books:
9
Input the information of books (name, date, state, subject):
book1 1990 0 Art
book2 1995 1 History
book3 1992 1 History
book4 2007 0 Science
book5 2017 1 Art
book6 1997 1 History
book7 2001 0 Science
book8 2014 1 Science
book9 2000 1 Art
Books in the library:
Art:
book1 1990 0
book9 2000 1
book5 2017 1
Science:
book7 2001 0
book4 2007 0
book8 2014 1
History:
book3 1992 1
book2 1995 1
book6 1997 1
Input the name of the book you want to borrow:
book1
Sorry, the book is not available. Try again:
book3
Succeed!
```