CSC3002 Complete Assignment 1

Important Notes:

1. The assignment is an individual project, to be finished on one's own effort.
2. **Submission deadline is 6 pm Friday, 2 February, 2024 (Week #4). After the deadline, we will reopen OJ for late submission for more 3 days.** Late submission will lead to some penalty – 10% deduction for each day of the delay and no submission would be accepted after 3 days of the due date.

3. Please also submit your final **four programs** and **your one page document**. The four programs should be renamed as "StudentID_A1_calculator.cpp", "StudentID_A1_tic_game.cpp", "StudentID_A1_rem_comments.cpp", and "StudentID_A1_ per_com.cpp", respectively on the blackboard. For example, a student whose Student ID is "120000001" should submit **four programs** named as "120000001_A1_ calculator.cpp", "120000001_A1_tic_game.cpp", "120000001_A1_ rem_comments.cpp", and "120000001_A1_per_com.cpp". Furthermore, prepare **one page document to highlight your ideas of your solutions to the given four problems**. The one page document should be named as "StudentID_A1_solutions.pdf".

4. OJ website: http://10.26.200.13/. After opening this website, go the contest "**CSC3002 24Spring Assingment 1"**. Please u**se your student ID** as the user name for registration. We score your assignment only using your student ID. If you are off campus, please use VPN to access the OJ.

5. Each student is only permitted to submit code to OJ **up to 80 times for each problem**. Only the **last submission** will be used in evaluation of assignment marks.

6. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% of marks will be deducted.

Marking Criterion:

1. The full score of the assignment is 100 marks.
2. **We have 4 problems in this assignment with 5 test cases each. Each test case has 5 marks.**
3. Zero mark is given if: there is no submission; a normal submission fails all test cases.

Running Environment:

1. The submissions will be evaluated in the course's OJ system running **C++17** and Linux platform.
2. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.
3. In the test, each program is required to finish within **5 seconds**, with no more than **64MB memory**. **This is a strict requirement measured in the server environment!**

Submission Guidelines:

1. Inconsistency with or violation from the guideline leads to marks deduction.

2. It is the students' responsibility to read this assignment document and submission guidelines carefully and in detail. No argument will be accepted on issues that have been specified clearly in these documents.
3.

Problem 1 Description (A Simple Calculator):

Write a program called **calculator.cpp** as a basic calculator to evaluate the value of some math expressions, and outputs an integer value for each expression.

We would have **5** test cases for evaluation and each test case is a valid math expression that takes up a line. For each test case, you should output its result as a line.

Input: Here is an illustration of the input math expression at a line.

-3+4/3+5

1. Each expression includes **only integer numbers** and contains no more than five operators of "+" (addition), "-" (subtraction), "*" (multiplication) and "/" (division).

2. Each expression has no blank space.

3. All five input expressions **are valid**.

Output: For the output, please output an **integer value after rounding** for each input expression. Note that no space can be allowed in the output result.

Example 1: the input a math expression is as follows:

-3+4/3+5

The corresponding output is as follows:

3

Example 2: the input a math expression is as follows:

1+2+4*5

The corresponding output is as follows:

23

Example 3: the input a math expression is as follows:

1-3*5-13/4

The corresponding output is as follows:

-17

Problem 2 Description (Check-Tic-Tac-Toe-Game)

Write a program called **tic_game.cpp** that checks whether there is a winner for the Tic Tac Toe game. The rule of the game is that two players alternately mark a 3×3 grid with an X or an O. The game is won by the person who successfully arranges three of their marks in a row that is either

horizontal, vertical, or diagonal. We would have **5** test cases for evaluation.

Input: The program reads **three** lines from the input, which is the 3×3 grid of the game. Each line has 3 characters which are "X", "O", or "E". The "E" character represents that the position is empty and no player has marked this position.

Output: Print "True" if the game ends, **which is either there is a winner or all positions in the grid are non-empty. Otherwise, Print "False".**

Here are the detailed rules of the Tic Tac Toe game:

1) Players take turns placing characters into empty positions.
2) The first player always places 'X' characters, while the second player always places 'O' characters.
3) 'X' and 'O' characters are always placed into empty positions, never filled ones.
4) The game ends when there are three of the same (non-empty) character filling any row, column, or diagonal.
5) The game also ends if all positions are non-empty.
6) No more moves can be played if the game is over.

Example 1: the input is as follows:

```
XOX
XOO
EXE
```

The corresponding output is as follows:

```
False
```

Example 2: the input is as follows:

```
XOE
EXO
EEX
```

The corresponding output is as follows:

```
True
```

Example 3: the input is as follows:

```
XOX
XOX
OXO
```

The corresponding output is as follows:

```
True
```

Problem 3 Description (Remove-Comments)

Write a program called **rem_comments.cpp** that remove comments from a C++ source code. Given a C++ code, remove comments from it. **The source code is a text with multiple lines of string**.
In C++, there are two types of comments, line comments, and block comments. We would have **5**

test cases for evaluation.

1) **The string "//"** denotes a **line comment**, which represents that it and the rest of the characters to the right of it in the same line should be ignored.
2) **The string "/*"** denotes a block comment, which represents that all characters until the next (non-overlapping) occurrence of **"*/" should be ignored**. (Here, occurrences happen in reading order: line by line from left to right.) To be clear, the string "/*/" does not yet end the block comment, as the ending would be overlapping the beginning.

- The first effective comment takes precedence over others. For example, if the string "//" occurs in a block comment, it is ignored. Similarly, if the string "/*" occurs in a line or block comment, it is also ignored.
- If a certain line of code is empty after removing comments, you must not output that line: each string in the answer list **will be non-empty.**
- There will be **no control characters, single quote, or double quote characters**. For example, source = "string s = "/* Not a comment. */";" **will not be a test case**. Also, nothing else such as defines or macros will interfere with the comments.
- **It is guaranteed that every open block comment will eventually be closed**, so "/*" outside of a line or block comment always starts a new comment.
- **It is guaranteed that there is no blank line in the source code.**
- Finally, implicit newline characters can be deleted by block comments. Please see the examples below for details.

After removing the comments from the source code, return the source code in the same format.

Notice: How to read multiple lines from console? Here we provide an example code as follows:

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

   int main()
{

   vector<string> source_v;
   std::string line;
   while(std::cin) {
      getline(std::cin,line);
      source_v.push_back(line);
   }
}
```

Example 1: the source code of input is as follows:

```
/*Test program */
int main()
{
  // variable declaration
```

```
int a, b, c;
/* This is a test
     multiline
     comment for
     testing */
a = b + c;
}
```

The corresponding output is as follows:

```
int main()
{
int a, b, c;
a = b + c;
}
```

Example 2: the input is as follows:

```
int a;
/*comment
line more_comment
*/
int b;
```

The corresponding output is as follows:

```
int a;
int b;
```

Example 3: the input is as follows:

```
#include <string>
#include <iostream>
#include <vector>
#include <cstdio>
#include <unordered_map>
#include <unordered_set>
#include <set>
#include <map>
#include <cstdlib>
#include <algorithm>
#include <queue>
// accumulate example
#include <functional>      // std::minus
#include <numeric>           // std::accumulate
using namespace std;
```

```
int main() {
	int n;
	int a;
	cin >> n;
	vector<int> nums;
	for (int i = 0; i < n; i++) {
		cin >> a;
		nums.push_back(a);
	}
	/* happy weekend
	   happy new year
	  Year of The Dragon */
	int max_len = 1;
	vector<int> res(n + 1, 0);
	res[0]=1;
	for (int i = 1; i < n; i++) {
		int temp_res = 0;
		for (int j = 0; j < i; j++) {
			if (nums[j] < nums[i]) {
				if (temp_res < res[j]) {
					temp_res = res[j];
				}
			}
		}
		res[i] = max(1, temp_res + 1);
		max_len = max(res[i], max_len);
	}
	cout << max_len << endl;
}
```

The corresponding output is as follows:

```
#include <string>
#include <iostream>
#include <vector>
#include <cstdio>
#include <unordered_map>
#include <unordered_set>
#include <set>
#include <map>
#include <cstdlib>
#include <algorithm>
#include <queue>
#include <functional>
```

```
#include <numeric>
using namespace std;
int main() {
	int n;
	int a;
	cin >> n;
	vector<int> nums;
	for (int i = 0; i < n; i++) {
		cin >> a;
		nums.push_back(a);
	}
	int max_len = 1;
	vector<int> res(n + 1, 0);
	res[0]=1;
	for (int i = 1; i < n; i++) {
		int temp_res = 0;
		for (int j = 0; j < i; j++) {
			if (nums[j] < nums[i]) {
				if (temp_res < res[j]) {
					temp_res = res[j];
				}
			}
		}
		res[i] = max(1, temp_res + 1);
		max_len = max(res[i], max_len);
	}
	cout << max_len << endl;
}
```

Problem 4 Description (Permutation and Combination)

Write a C++ Program called **per_com.cpp** to implement the mathematical formula of permutation and combination and compute the numbers of them for two input integers, n and r. The mathematical formula of permutation and combination are as follows. We would have **5** test cases for evaluation.

Please use the **recursion** to implement the formula. **Recursion** is the technique of making a function call itself. You can learn it via https://www.w3schools.com/cpp/cpp_functions_recursion.asp.

$$Permutation(n,r) = \frac{n!}{(n-r)!}$$

$$Combination(n,r) = \frac{n!}{r!\,(n-r)!}$$

Input: The program reads two integers from a line. The first integer is n and second is r where 1<=n<=15, and 1<=r<=15.

Output: The output has two lines. The first line is the number of permutation with prefix 'Permutation: 'and the second line is the number of combination with prefix 'Combination: '. Please see the example below.

Example 1: the input is as follows:

```
5 3
```

The corresponding output is as follows:

```
Permutation: 60
Combination: 10
```

Example 2: the input is as follows:

```
14 5
```

The corresponding output is as follows:

```
Permutation: 240240
Combination: 2002
```

Example 3: the input is as follows:

```
6 1
```

The corresponding output is as follows:

```
Permutation: 6
Combination: 6
```