RC.CODE.C++

1 Basic information

1.1 Competition content

A. he competition will comprehensively test the contestants' technical implementation ability based on C++ software programming language, encourage innovation, cultivate practical skills, and problem-solving abilities.

B. The competition are participated by individuals.

C. The competition items are set with objective questions and programming questions.

D. All groups in the competition participate in answering questions on the designated platform using a computer.

E. The duration of each competition is 90 minutes.

1.2 Group setting

Programming event	programming language	Group division	Age requirement
C++ programming	C++	C++ - A	≤12 years old
	C++	C++ - B	≤18 years old

1.3 Competition topic

Group	Single Choice Question	Programming Questions	Competition time
C++ - A	5 questions totaling 300 points	5 questions, totaling 300 points	90 Min
C++ - B	5 questions totaling 300 points	5 questions, totaling 300 points	90 Min

2 Rules and scores

2.1 Competition rules

A. A. The competition requires contestants to use coding methods on the designated platform to complete the designated questions. During the answering process, it is prohibited to cut out the compiler or open other software and web pages. Otherwise, it will be considered cheating and the score will be cancelled.

B. Each contestant has only one chance to challenge, and those who fail to log in to the designated platform within the specified time will be considered as giving up the challenge.

C. In the preparation stage, contestants must complete the answer preparation according to the designated platform's prescribed steps and independently operate to enter the formal answer.

D. During the competition period, contestants are not allowed to leave the computer answering area.

E. During the competition period, contestants are not allowed to plagiarize others, cheat, or directly contact other contestants' computers. If there is a violation, the contestant will receive 0 points.

F. During the competition process, one is not allowed to seek help from others, interfere with other contestants' preparation and answering questions, or damage public equipment.

G. During the competition period, no communication software may be opened on the computer. Any violation will result in 0 points for the contestant.

H. During the competition period, contestants are not allowed to use communication, photography electronic devices such as mobile phones, phone watches, and external storage devices. Any violation will result in disqualification from the competition.

I. The interpretation of these rules belongs to the RC Organizing Committee.

2.2 Competition score

According to the completion status of the questions, as well as the comprehensive evaluation of completion degree and time, the more questions completed, the higher the completion degree, and the shorter the time, the higher the score of the contestants.

3 Equipment requirements

A. Bring your own computer, computer operating system: Mac OS, Windows 10 or above operating system; The browser should use Google Chrome (version 69.0 or above), Firefox, Internet Explorer 11 or above, and Chrome is recommended.

B. Bring your own smartphone.

4 Outline requirements

4.1 C++ - A requirements

A. Programming Fundamentals:

a. Sequential Structure: Understand program flow, basic input and output.
   a) Master the definition and use of variables.
   b) Master basic operation statements and input/output statements.
   c) Able to complete programs with sequential structure.

b. Branching Structure: if conditional statements, simple logical operations.
   a) Understand the basic concepts of logical operations.
   b) Master the function and writing of basic logical expressions.
   c) Able to complete programs with selection structure.
   d) Master the writing of the ternary operator.

c. Loop Structure: for loops, while loops to solve repetitive tasks.
   a) Master the function and writing of basic loop statements.
   b) Able to complete programs with loop structure.

d. Arrays: Using arrays to store and access data collections.
   a) Understand the concept of arrays.
   b) Master the definition, assignment, and query methods of arrays.

e. Strings: Basics of string operations, such as concatenation, searching for characters, etc.
   a) Master the use of character array-related functions.

B. Mathematical Knowledge:

a. Algebra: Addition, subtraction, multiplication, and division of integral expressions.
   a) Understand common mathematical functions and master their usage.

b. Geometry: Understand the representation of points and line segments within a coordinate system.
   a) Perform simple geometric figure operations through programming.

c. Functions: Understand linear functions and their graphs.
   a) Initially master the use of mathematical library functions.

C. Algorithms:

a. Simulation: Directly implement functions according to the problem description.
   a) Understand the concept of the simulation method.
   b) Able to use the simulation method to solve relatively simple problems.

b. Enumeration: Use loops to exhaust possibilities to find the answer.
   a) Understand the concept of the enumeration method.
   b) Able to use the enumeration method to solve relatively simple problems.

4.2 C++ - B requirements

A. Programming Fundamentals:

a. Sequential Structure: Understand program flow, basic input and output.

- a) Master basic file read and write operations.
- b) Master type conversion of variables.

b. Branching Structure: if conditional statements, simple logical operations.

- a) Master the use of multi-level branch structures.
- b) Able to write programs with multi-level branch structures.

c. Loop Structure: for loops, while loops to solve repetitive tasks.

- a) Master the use of multi-level loop structures.
- b) Able to write programs with multi-level loop structures.

d. Arrays: Using arrays to store and access data collections; advanced applications of multidimensional arrays.

- a) Able to solve comprehensive problems involving branch structures, loop structures, arrays, etc.

e. Strings: Basics of string operations, such as concatenation, searching for characters, etc. Advanced string processing, including advanced operations like substring extraction.

- a) Master the advanced use of character array-related functions.

f. Structure definition and use.

- a) Understand the concepts of structures and classes.
- b) Master the use of structures and classes.

g. Branching and loop structures involving more complex logical judgments and nested use.

h. Multi-keyword sorting and techniques for deduplication sorting.

- a) Master various sorting methods including Bucket Sort, Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, etc.

i. Strengthening the concepts of custom functions and recursive calls; introduction to file operations.

- a) Understand the concept of functions.
- b) Master the definition and calling of functions.
- c) Understand the concept of function parameter passing.
- d) Master the methods of function parameter passing.
- e) Master the concept and method of recursion.
- f) Master basic file read and write operations.

B. Data Structures:

a. set/map/pair: Master the associative containers set/map and the data pair 'pair'.

- a) Know about set and map in the STL.
- b) Master the use of set and map in the STL.

b. Stack/Queue: Use the standard library's stack/queue to complete specific tasks.

a) Master the use of stacks and queues.

c. Linked Lists: Creation and traversal of basic linked list nodes.

b) Master the use of various linked lists (singly linked lists, doubly linked lists, circular linked lists, etc.).

C. Mathematical Knowledge:

a. Functions: Including quadratic functions and inverse proportional functions.

a) Implement function graph plotting and calculation through programming.

b. Equations: Solving quadratic equations and applied problems of equations.

a) Master the extended Euclidean algorithm and use it to solve congruence equations.

b) Master Gaussian elimination and use it to solve problems.

c. Combinatorial Counting

a) Master the Addition Principle and Multiplication Principle, and use them to solve simple problems.

b) Understand the definitions of permutations and combinations.

c) Master the calculation formulas for permutations and combinations.

D. Algorithms:

a. Simulation: Directly implement functions according to the problem description.

a) Able to use the simulation method to solve relatively complex problems.

b. Enumeration: Use loops to exhaust possibilities to find the answer.

a) Able to use the enumeration method to solve relatively complex problems.

c. Introduction to High-Precision Operations.

a) Master the four arithmetic operations for high-precision numbers.

d. Application of Divide and Conquer.

a) Master the binary search method and use it to solve relatively simple problems.

b) Understand the idea of the multiplication method and use it to solve problems.

e. Simple Application of Greedy Algorithms.

a) Master simple greedy algorithms and use them to solve relatively simple problems.

b) Able to design more complex greedy algorithms and use them to solve problems.

f. Sorting algorithms including but not limited to Merge Sort and Quick Sort.

   a) Master sorting methods such as Quick Sort and Merge Sort.

g. Search Algorithms and Their Pruning.
   a) Master Depth-First Search and Breadth-First Search.
   b) Master various search pruning methods.

h. Dynamic Programming Algorithms.
   a) Understand the concept of dynamic programming.
   b) Able to use the knapsack algorithm to solve relatively simple problems.
   c) Able to use interval dynamic programming to solve relatively simple problems.
   d) Able to design relatively simple dynamic programming and use it to solve problems.
   e) Able to design relatively complex dynamic programming and use it to solve problems.
   f) Master dynamic programming on complex structures like trees and state compression, and use it to solve problems.
   g) Master state design methods for complex dynamic programming and use them to solve problems.
   h) Master complex dynamic programming optimization methods like slope optimization and decision monotonicity optimization, and use them to solve problems.

i. Number Theory Algorithms.
   a) Master the Euclidean algorithm for finding the greatest common divisor.
   b) Master the Sieve of Eratosthenes and the Linear Sieve for prime numbers.
   c) Master the basic properties of modular arithmetic.
   d) Understand the concept of modular inverses.
   e) Master Fermat's Little Theorem and use it to find modular inverses.
   f) Master Euler's Theorem and use it to solve problems.
   g) Master the Extended Euler's Theorem and use it to solve problems.
   h) Master the Chinese Remainder Theorem and use it to solve systems of congruence equations.

j. Graph Theory Algorithms.
   a) Master the definition, storage, and traversal of graphs.
   b) Understand the idea of topological sorting and master topological sorting on directed acyclic graphs.
   c) Master various shortest path algorithms.
   d) Master various minimum spanning tree algorithms.
   e) Understand concepts like connected components, strongly connected graphs, and strongly connected components.

   f) Master the method for finding strongly connected components in directed graphs.
   g) Master the method for finding cut vertices and bridges in undirected graphs.
   h) Master the method for finding biconnected components and edge biconnected components in undirected graphs.
   i) Master the idea of vertex contraction and use it to solve problems.

k. String Algorithms.
   a) Master the KMP algorithm for string matching.

l. Advanced Data Structures.
   a) Master the use of the Sparse Table.
   b) Master the use of the Disjoint Set Union.
   c) Master the use of the Fenwick Tree.
   d) Master the use of the Segment Tree.
   e) Master the use of the Trie tree.
   f) Master the use of at least one type of balanced binary search tree, including but not limited to treap, splay tree, etc.

m. Other Algorithms.
   a) Master the Fast Exponentiation algorithm.
   b) Master simple recurrence algorithms and use them to solve relatively simple problems.
   c) Master the prefix sum algorithm and use it to solve relatively simple problems.
   d) Master the difference algorithm and use it to solve relatively simple problems.
   e) Master the two-pointer technique and use it to solve relatively simple problems.
   f) Understand the concept of hashing algorithms and able to design relatively simple hash functions.
   g) Master the usage of hash tables.
   h) Understand the concept of Eulerian circuits.
   i) Master the method for finding Eulerian circuits.
   j) Master the Inclusion-Exclusion Principle and use it to solve problems.
   k) Understand the structure and basic properties of linear basis and use it to solve problems.

5 RC.CODE.C++ Sample question example

【C++ - A 】Sample question example

一、Multiple choice questions (20 points each)

1、The correct expression among the following is ( )

A. 7++

B. (a+b)++

C. ++(a+b)

D. ++x

Answer D

2、If we define int a=2, b=2, the value of the following expression is 6 ( )

A. a*(++b)

B. a*(b++)

C. a+b

D. a*b

Answer A

3、When t is of type int and before entering the following loop, the value of t is 0. Which of the following statements is correct? ( )

```
while(t=1){...}
```

A. The value of the loop condition expression is 0

B. The value of the loop condition expression is 1

C. The loop condition expression is illegal

D. None of the above is correct

Answer B

4、Among the following 4 numbers expressed in different bases, the one with the largest value is ( )

A. Octal number 52

B. Decimal number 44

C. Hexadecimal number 2F

D. Binary number 101000

Answer C

5、The structure that organizes data according to the "first in, last out" principle is（　） L = []

A. Queue

B. Stack

C. Doubly linked list

D. Binary tree

Answer B

二、Programming questions (total score 300 points)

1、Programming Question 1（40 point）

Programming Implementation：Sorting

Title description:

Input three positive integers and output them in ascending order.

Input description:

Input three positive integers separated by a space

Output description:

Output three positive integers in ascending order and separated by a space

Sample input:

15 8 10

Sample output:

8 10 15

Scoring Criteria:

10 point：correctly output a set of data;

10 point：correctly output two sets of data;

10 point：correctly output three sets of data;

10 point：correctly output four sets of data;

2、Programming Question 2（40 point）

Programming Implementation：freight

Title description:

When taking a plane, if the luggage exceeds the specified weight, the luggage will be checked in and a check-in fee will be charged.

The following is the luggage check-in fee of a certain airline:
The luggage weight is within 20 kg (including 20) and is charged at 10 yuan per kilogram. The part exceeding 20 kg is charged at 15 yuan per kilogram.
Please write a program to calculate the fee for checked luggage given the total weight of the luggage (unit: kg).

Input description:
Enter a positive integer N (5<N<200) as the total weight of the luggage (unit: kg)
Output description:
Output the fee for checking N kg of luggage

Sample input: 10
Sample output: 100

Scoring Criteria:

10 point: correctly output a set of data;

10 point: correctly output two sets of data;

10 point: correctly output three sets of data;

10 point: correctly output four sets of data;

3、Programming Question 3 (60 point)

Programming Implementation: Find the sum of prime numbers
Title description:
Given a positive integer N, calculate the sum of all prime numbers between 2 and N.
For example: If N is equal to 10, the prime numbers between 2 and N are 2, 3, 5, and 7, and the sum of all prime numbers is equal to 17 (2 + 3 + 5 + 7).

Input description:
Input a positive integer N (2<N<101)
Output description:
Output an integer representing the sum of all prime numbers between 2 and N (including 2 and N)

Sample input: 10

Sample output: 17

Scoring Criteria:

15 point: correctly output a set of data;

15 point: correctly output two sets of data;

15point: correctly output three sets of data;

15 point: correctly output four sets of data.

4、Programming Question 4 (60 point)

Programming Implementation: Find the length of descending subsequence

Title description:

There is a set of integer sequences of length N. Find the longest descending subsequence containing the Kth integer from the sequence and output the length of the subsequence (a descending subsequence means that the numbers in the sequence are decreasing from left to right, that is, the number on the right is smaller than the number on the left, and if they are equal, they are not considered decreasing).

For example: the integer sequence of length 5 is [4, 6, 2, 4, 8], and K is 2.
The second integer is 6, and the longest descending subsequence containing 6 is [6, 4, 2], so the length is 3.

Input description:

In the first line, input two positive integers N (2<N<100) and K (0<K≤N), which represent the length of the integer sequence and the number of elements to be included respectively

In the second line, input N integer sequences (−1000<integer<1000), with a space between integers

Output description:

Output the length of the longest descending subsequence containing the Kth element

Sample input:

```
5 2
4 6 2 4 8
```

Sample output:

3

Scoring Criteria:

15 point: correctly output a set of data;

15 point: correctly output two sets of data;

15point: correctly output three sets of data;

15 point: correctly output four sets of data.

5、Programming Question 5 (100 point)

***Programming Implementation:* Minimum**

Title description:

Given a string containing only numbers, with a length of N (5<N≤20), and a positive integer M (1≤M≤5). Use M multiplication signs to insert into the string, and the two multiplication signs cannot be adjacent. After insertion, a multiplication formula is generated. Find a way to insert the multiplication formula with the smallest value, and output the result. (The multiplication sign cannot be placed at the beginning or end of the string)

Note:

Insertion position: The multiplication sign can be inserted between numbers, but not at the beginning or end of the string, that is, the multiplication sign can only be placed between two numbers.

Adjacent multiplication signs: The inserted multiplication signs cannot be adjacent, which means that there must be at least one number between the two multiplication signs.

For example, the string is 123456, and 2 multiplication signs are inserted.

The insertion methods are:

1×2×3456=6912

1×23×456=10488

1×234×56=13104

1×2345×6=14070

12×3×456=16416

12×34×56=22848

12×345×6=24840

123×4×56=27552

123×45×6=33210

1234×5×6=37020

Among them, the second multiplication formula has the smallest value, which is 6912.

Input description:

In the first line, enter two positive integers N (5<N≤20) and M (1≤M≤5), where N represents the length of the string and M represents the number of multiplication signs. There is a space between the two positive integers.
In the second line, enter a string of length N that contains only numbers, indicating that a string with M multiplication signs is to be inserted.

Output description:

Output an integer representing the minimum result.

Sample input:

6 2
123456

Sample output:

6912

Scoring Criteria:

25 point: correctly output a set of data;

25 point: correctly output two sets of data;

25point: correctly output three sets of data;

25 point: correctly output four sets of data.

【C++ - B】Sample question example

一、Multiple choice questions (20 points each, total score 100 points)

1、Which of the following is not a basic data type in C++? (   )

A. int
B. float
C. string
D. char

Answer C

2、In C++, which symbol is used to represent a comment? ( )

A. //
B. /* */
C. #
D. Both A and B are correct
Answer D

3、Which of the following options is a standard library function used for output in C++? ( )
A. input()
B. print()
C. cout
D. output()
Answer C

4、Which of the following statements about C++ structures is correct? ( )

A structure can only contain member variables, not member functions
B structure cannot inherit from another structure
C structure can contain static member variables
D structure cannot contain constructors
Answer C

5、Execute the following code, the output result is ( )

```
#include <iostream>
using namespace std;
int f(int k)
{
    if (k <= 2)
    {
        return 1;
    }
    return 2 * f(k - 2) + f(k - 1);
}
int main()
{
    int n = 7;
    cout << f(n);
    return 0;
}
```

A. 21
B. 41
C. 43
D. 45
Answer C

二、Programming questions (total score 300 points)

1、Programming Question 1 (40 point)

Programming Implementation:: Find the number of digits

Title description:
Given a positive integer N (1<N<200), output how many digits N is.

Input description:
Input a positive integer N (1<N<200)
Output description:
Output an integer indicating how many digits N is

Sample input: 11
Sample output: 2

Scoring Criteria:

10 point: correctly output a set of data;

10 point: correctly output two sets of data;

10 point: correctly output three sets of data;

10 point: correctly output four sets of data.

2、Programming Question 2 (40 point)

Programming Implementation: Spelling Words

Title description:

The English words for four kinds of fruits are Apple, Banana, Cherry, and Date. The teacher prompts the first letter of each word and asks the students to spell the corresponding word.
Please write a program:
When the uppercase letter input is "A", it outputs "Apple";
When the uppercase letter input is "B", it outputs "Banana";
When the uppercase letter input is "C", it outputs "Cherry";
When the uppercase letter input is "D", it outputs "Date".

Input description:
Enter any uppercase letter from A, B, C, D
Output description:
Output a string representing the English word corresponding to the uppercase letter (the first letter of the word is capitalized)
Sample input: A
Sample output: Apple

Scoring Criteria:

10 point: correctly output a set of data;

10 point: correctly output two sets of data;

10 point: correctly output three sets of data;

10 point: correctly output four sets of data.

3、Programming Question 3 (60 point)

Programming Implementation: Judging numbers
Title description:
Given a positive integer N (100≤N＜100000), count the number of positive integers between 100 and N (including 100 and N) that meet the following

conditions:
1) The unit digit of the positive integer is not 3;
2) The tens digit of the positive integer is not 5;
3) The hundreds digit of the positive integer is not 7.

Input description:
Input a positive integer N (100≤N＜100000)

Output description:

Output how many positive integers between 100 and N (including 100 and N) meet the conditions

Sample input: 110

Sample output: 10

Scoring Criteria:

15 point: correctly output a set of data;

15 point: correctly output two sets of data;

15point: correctly output three sets of data;

15 point: correctly output four sets of data.

4、Programming Question 4 (60 point)

Programming Implementation: Calculation 24

Title description:

"Calculate 24" is a classic number game.

The rules of the game are: pick out 4 numbers from the natural numbers between 1 and 10 (the 4 numbers are different and the order is random), perform addition, subtraction, and multiplication (the number of times and types of a certain operation are not limited), and the result of the operation must be equal to 24. Multiplication has a higher priority than addition and subtraction, and brackets cannot be used in the formula, and the order of the numbers must remain unchanged.

Example:

If the 4 numbers given are: 10, 2, 4, 8, there are two solutions (10+2+4+8=24, 10*2-4+8=24), then output 2

If the 4 numbers given are: 7, 2, 3, 6, there are zero solutions, then output 0

Input description:

Input four positive integers between 1 and 10 and separate them with a space (the four positive integers are different)

Output description:

Output how many operation solutions have the result of 24

Sample input: 10 2 4 8

Sample output: 2

Scoring Criteria:

15 point: correctly output a set of data;

15 point: correctly output two sets of data;

15point: correctly output three sets of data;

15 point: correctly output four sets of data.

5、Programming Question 5 (100 point)

Programming Implementation:

Title description:

There are n cities (numbered 1 to n) and m two-way roads in a certain region. Each road connects two cities and has a travel time (positive integer). Some cities are equipped with "charging stations".

An electric vehicle starts from the starting point s and aims to reach the destination t. The vehicle has a battery capacity of C (i.e., it can travel at most C units of time on a full charge), and the battery is initially fully charged.

Rules:

For each road with travel time w, the battery level decreases by w.

If the vehicle arrives at a city with a charging station, it may choose to fully recharge the battery (or choose not to recharge).

Find a path from s to t such that the total travel time is minimized. If no feasible path exists, output -1.

Input description:

First line: Five integers n, m, s, t, C ($2 \le n \le 50$, $1 \le m \le 200$, $1 \le s, t \le n$, $s \ne t$, $1 \le C \le 100$).

Second line: n integers (0 or 1), representing is_charge[1] to is_charge[n].

Next m lines: Each line contains three integers u, v, w, indicating a road between u and v with travel time w ($1 \le w \le C$).

Output description:

A single integer representing the shortest total travel time; if it is unreachable, output -1.

Sample input:

```
5 5 1 5 10
0 1 0 1 0
1 2 5
2 3 3
3 4 4
4 5 2
2 4 6
```

Sample output:

```
13
```

Scoring Criteria:

25 point: correctly output a set of data;

25 point: correctly output two sets of data;

25point: correctly output three sets of data;

25 point: correctly output four sets of data.