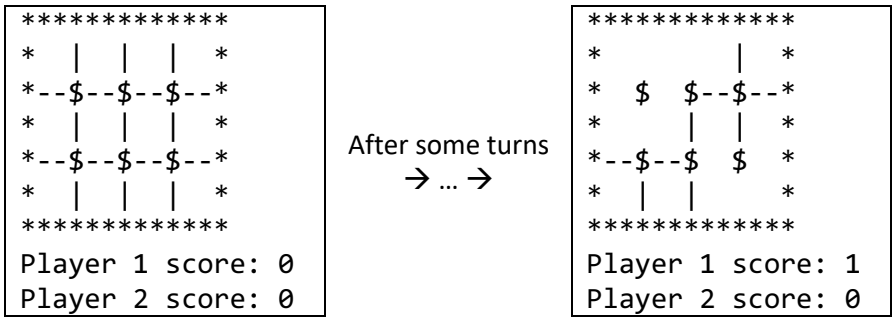
Assignment 3: Strings and Coins

Due: 23:59, Thu 20 Oct 2022 File name: `stringcoin.cpp` `stringcoin-game.cpp` `stringcoin.h` Full marks: 100

Introduction

The objective of this assignment is to practice (1) defining functions (being a callee), (2) calling functions (being a caller), and (3) representing special kind of data.

You will implement a game called *Strings and Coins*. At the beginning of the game, there is a network of coins, each connected by four strings. Two players take turns cutting a string in the network. When a cut leaves a coin with no strings connected, the player scores one point *and* takes an extra turn. Note that the player can get more than two successive turns as long as s/he keeps scoring in every turn. The game ends when all strings are cut, and the player with more points wins. It is a draw if two players get the same points. Figure 1 shows an example game network. We use the symbol `$` to denote coins, `--` and `|` to denote the strings, and `*` to denote the "wall" (that is, the network boundary). In Figure 1(b), the top-left coin is already disconnected (assumed to be by Player 1), and the bottom-right coin is one string from being disconnected.



(a) (b)

Figure 1: (a) Initial Game Network; (b) Network after some turns

Program Specification

This section describes the game network representation, some necessary functions, and program flow.

Basic Requirements

- You *cannot declare any global variables* (variables declared outside any functions) in all files.
- You *cannot use any functions in the* `<cmath>` library.
- You *cannot use any arrays nor vectors.*

Game Network Representation

There are 17 strings initially in the game network. Therefore, we use integers 1 to 17 to denote these string positions, as illustrated in Figure 2. The positions are basically ordered top-to-bottom, left-to-right.

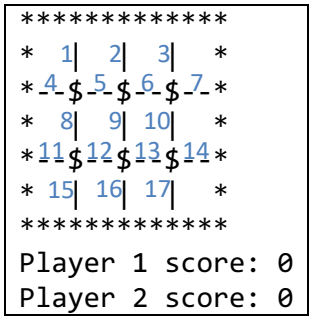


Figure 2: Numbers for String Positions

To encode the whole network and the players' scores in a game, we use a 19-digit integer $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}d_{11}d_{12}d_{13}d_{14}d_{15}d_{16}d_{17}d_{18}d_{19}$. The first 17 digits $d_1 \ldots d_{17}$ are either 0 or 1, denoting whether the string in the corresponding positions 1 to 17 are cut or not. (That is, $d_i = 0$ means the string in position i is already cut; $d_i = 1$ means position i remains uncut.) The last two digits d_{18} and d_{19} range in 0–6 to denote the scores of Players 1 and 2 respectively. For example, the network in Figure 1(b) is encoded by 00100011011110011010, which also stores the scores of Player 1 (1) and Player 2 (0). Using this representation, the initial full network (with all strings uncut) is encoded as the integer 11111111111111111100. The network at game end (with all strings cut) is encoded as 00000000000000000*xy*, where *x* and *y* depend on the scores of Players 1 and 2. For example, 00000000000000000024 is a game won by Player 2. (Note: in C++, integer constants should *NOT contain leading zeroes*, otherwise, they will be treated as octal numbers. Therefore, the network in Figure 1(b), e.g., is actually 10011011110011010 rather than 00100011011110011010.)

The data type `int` in C++ is typically 32-bit and is not big enough to store a 19-digit integer. In your program, you have to use a bigger integer type called `long long`. In Visual Studio, `long long` is a 64-bit signed integer type, whose range is -9,223,372,036,854,775,808 … 9,223,372,036,854,775,807. We can simply use one `long long` variable to store a game network.

Provided and Required Functions (`stringcoin.cpp` and `stringcoin.h`)

Your program must contain the following functions. *Some of them are written for you already (Provided) and you shall not modify their contents.* The others will be written by you (Required). These functions shall be implemented in a source file `stringcoin.cpp` with the function prototypes in a header file `stringcoin.h`. These functions shall be called *somewhere* in your program. You must *not* modify the prototypes of all these functions. You can design extra functions if you find necessary.

In the functions below, you can assume that (a) the parameter `network` is always a proper encoding of a game network (19-digit integer; 1st–17th digits are 0 or 1; 18th and 19th digits are the scores, etc.); (b) the parameter `pos` is always 1–17; and (c) the parameter `p` is always either 1 or 2.

Besides, *all the required functions below will be graded individually.* That is, we shall connect your `stringcoin.cpp` with another source file with our testing code to call the functions one by one for grading. So your code for each function shall implement the description of that function only. You shall *not* write any code in a function that is beyond that function's description.

(Required) `bool stringState(long long network, int pos)`

Returns `true` if position `pos` of the game `network` still has a string (not cut yet); returns `false` otherwise (that is, string was already cut). You shall *not print anything* using `cout` in this function.

(Provided) `void printNetwork(long long network)`

Prints the `network` to the screen using the format in Figure 1.

(Required) `int playerScore(long long network, int p)`

Returns the score of Player `p` in `network`. (Either the 18th or 19th digit in `network`.) You shall *not print anything* using `cout` in this function.

(Required) `void updateNetwork(long long &network, int pos, int p)`

Performs the task of Player `p` cutting a string in position `pos` of `network`. The *reference parameter* `network` should get updated, and if any coins are disconnected, the score of Player `p` shall be incremented, to reflect the new network configuration. You can assume that position `pos` of `network` is always 1 (so that the string can be cut). The following table shows some sample function calls and the expected results.

Parameter `network`	Parameter `pos`	Parameter `p`	Value of `network` *after* calling `updateNetwork(network, pos, p)`
100110111110011010	10	2	1001101011001101**1**
1010101001001101110	5	1	1010001001001101**1**20
1010101001001101110	5	2	1010001001001101**1**11
1001011100110010001	11	1 or 2	100101110001001000**1**
1001001001000010002	15	1	10010010010000000**1**2
10000000000031	6	2	3**3**

E.g., the first sample call in the table is equivalent to Player 2 cutting a string in position 10 of the network in Figure 1(b), resulting in the coin in the bottom-right to be disconnected and thus Player 2 scoring one point (denoted by red color). Note that cutting one string can disconnect at most two coins. (See the last row in the table.) To determine if a new coin is disconnected, calling the `stringState(…)` function is useful. Besides, you shall *not print anything* using `cout` in this function.

Program Flow (`stringcoin-game.cpp`)

The program flow of the game is described as follows. You should call the functions above to aid your implementation.

1. The program starts with a full network (1111111111111111100). Player 1 takes the first turn.
2. Then, prompt the current player to enter a position to where s/he wants to cut a string. You can assume that the input is always an integer.
3. A user input is invalid if the input position is not 1–17, or the position was already cut. In case it is invalid, display a warning message and go back to Step 2.
4. Update the network by cutting the string in the input position.
5. If the current player has disconnected one or more coin(s), print a message and keep him/her the current player. Otherwise, swap the other player to become the current player.
6. Repeat steps 2–5 until all 17 strings have been cut. (That is, until game is over.)
7. Once all strings have been cut, determine the winner or a draw, and display the message *"Player 1 wins!"*, *"Player 2 wins!"*, or *"Draw game!"* accordingly.

Functions and Files Organization

- The file `stringcoin.cpp` must contain the implementation of the provided and required functions. There must be no `main()` function in the file. You may write extra functions in this file if you find necessary.
- The header file `stringcoin.h` must contain the prototypes of the provided and required functions. You may put in extra function prototypes if you have written extra functions in `stringcoin.cpp`.
- The file `stringcoin-game.cpp` must contain the `main()` function for the program flow. You may write extra functions in this file if you find necessary.

Sample Run

The following shows a sample run. The blue text is user input and the other text is the program printout. You can try the provided sample program for other input. *Your program output should be exactly the same as the sample program* (same text, symbols, letter case, spacings, etc.). Note that there is a space after ':' in the printout.

```
*************
*   |   |   |   *
*--$--$--$--*
*   |   |   |   *
*--$--$--$--*
*   |   |   |   *
*************
Player 1 score: 0
Player 2 score: 0
Player 1's move (1-17): 1↵
*************
*       |   |   *
*--$--$--$--*
*   |   |   |   *
*--$--$--$--*
*   |   |   |   *
*************
Player 1 score: 0
Player 2 score: 0
Player 2's move (1-17): -1↵
Invalid. Try again!
Player 2's move (1-17): 18↵
Invalid. Try again!
Player 2's move (1-17): 1↵
Invalid. Try again!
Player 2's move (1-17): 16↵
```

```
*************
*     |  |  *
*--$--$--$--*
*  |  |  |  *
*--$--$--$--*
*  |     |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 1's move (1-17): 2↵
*************
*        |  *
*--$--$--$--*
*  |  |  |  *
*--$--$--$--*
*  |     |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 2's move (1-17): 15↵
*************
*        |  *
*--$--$--$--*
*  |  |  |  *
*--$--$--$--*
*        |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 1's move (1-17): 4↵
*************
*        |  *
*  $--$--$--*
*  |  |  |  *
*--$--$--$--*
*        |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 2's move (1-17): 13↵
*************
*        |  *
*  $--$--$--*
*  |  |  |  *
*--$--$  $--*
*        |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 1's move (1-17): 8↵
```

```
*************
*         |  *
*  $--$--$--*
*     |  |  *
*--$--$  $--*
*         |  *
*************
Player 1 score: 0
Player 2 score: 0
Player 2's move (1-17): 5↵
Player 2 scores! Gets extra turn.
*************
*         |  *
*  $  $--$--*
*     |  |  *
*--$--$  $--*
*         |  *
*************
Player 1 score: 0
Player 2 score: 1
Player 2's move (1-17): 9↵
*************
*         |  *
*  $  $--$--*
*         |  *
*--$--$  $--*
*         |  *
*************
Player 1 score: 0
Player 2 score: 1
Player 1's move (1-17): 12↵
Player 1 scores! Gets extra turn.
*************
*         |  *
*  $  $--$--*
*         |  *
*--$  $  $--*
*         |  *
*************
Player 1 score: 1
Player 2 score: 1
Player 1's move (1-17): 11↵
Player 1 scores! Gets extra turn.
*************
*         |  *
*  $  $--$--*
*         |  *
*  $  $  $--*
*         |  *
*************
Player 1 score: 2
Player 2 score: 1
```

```
Player 1's move (1-17): 6↵
Player 1 scores! Gets extra turn.
*************
*         |  *
*  $  $  $--*
*         |  *
*  $  $  $--*
*         |  *
*************
Player 1 score: 3
Player 2 score: 1
Player 1's move (1-17): 10↵
*************
*         |  *
*  $  $  $--*
*            *
*  $  $  $--*
*         |  *
*************
Player 1 score: 3
Player 2 score: 1
Player 2's move (1-17): 3↵
*************
*            *
*  $  $  $--*
*            *
*  $  $  $--*
*         |  *
*************
Player 1 score: 3
Player 2 score: 1
Player 1's move (1-17): 7↵
Player 1 scores! Gets extra turn.
*************
*            *
*  $  $  $  *
*            *
*  $  $  $--*
*         |  *
*************
Player 1 score: 4
Player 2 score: 1
Player 1's move (1-17): 17↵
*************
*            *
*  $  $  $  *
*            *
*  $  $  $--*
*            *
*************
Player 1 score: 4
Player 2 score: 1
```

```
Player 2's move (1-17): 14↵
*************
*           *
*  $  $  $  *
*           *
*  $  $  $  *
*           *
*************
Player 1 score: 4
Player 2 score: 2
Player 1 wins!
```

Submission and Marking

- Your program file name should be `stringcoin.cpp`, `stringcoin-game.cpp`, and `stringcoin.h`. Submit the *three files* in Blackboard (https://blackboard.cuhk.edu.hk/). If you do not submit the `.h`, we shall assume that it is the same as the provided one.
- Insert *your name*, *student ID*, and *e-mail* as comments at the beginning of all your files.
- You can submit your assignment multiple times. Only the latest submission counts.
- Your program should be *free of compilation errors and warnings*.
- Your program should *include suitable comments as documentation*.
- ***Do NOT plagiarize.*** Sending your work to others is subjected to the same penalty as the copier.