EE1003 Introduction to Computer I

Programming Assignment 2: X-Linked Rings Puzzle

Due Date: 2022/11/16(Wed) 23:59:59

Background

X-Linked Ring is a well-known puzzle game for children, and many people spent a lot of effort to solve it when they were young. Figure 1 gives an example of the nine-linked ring. I don't know if you have discovered the laws of the linked rings in the process of playing? Just like the Fibonacci Series we have learned, it is easy to solve the X-linked ring problem by recursion. Specifically, it constantly runs around rules, and finally it can successfully separate its "ring" and "sword". On the other hand, after separation, how to restore it? In fact, as long as you follow its rules, you can restore it after repeated executions.

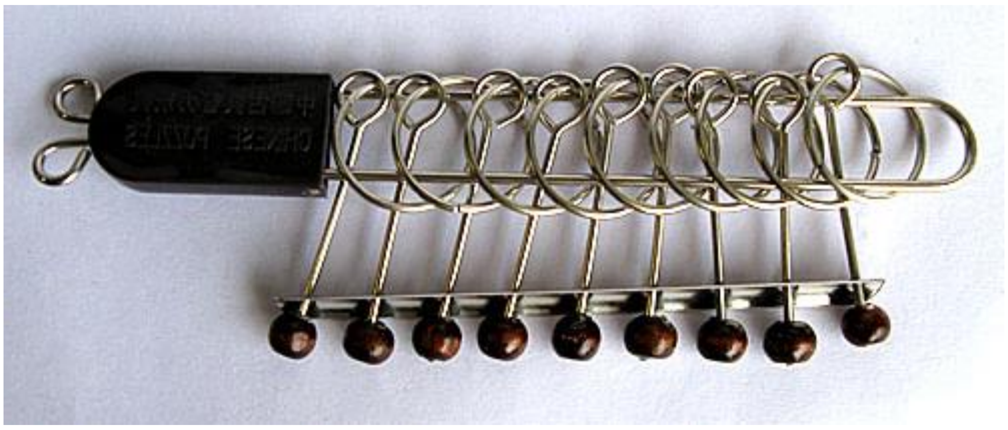

Figure 1. A nine linked ring

The rules are actually very simple. Suppose now there are two rules. The first rule is **R-rule**, which will move the outermost ring (rightmost ring in Figure 1) out of or on the sword. The second rule is **S-rule**, which will move the left ring of the rightmost ring on the sword. Here we denote a ring on the sword as 1 and a ring out of the sword as 0. Therefore, an unsolved x-linked ring can be represented as [111…111] from the inside to the outside (left to right in Figure 1). Then the problem is converted to the question "How to use the rules of R or S to make [111…111] into [000…000]". In this problem, you are asked to write a **recursion program** to simulate the way computer solve the x-linked ring.

Description

In this programming assignment, you are asked to write a program for simulating the solving procedure for de-attaching all rings of a x-linked ring from the sword with an arbitrary initial state. When the program starts, a welcome message is shown and ask the user to enter an integer *x* for the number of rings the program need to address. After that, the program will ask user to enter the current state of each ring from inside to the outside. If a ring is on the sword, the state is 1. If a ring is out of the sword, the state is 0. Then, the program will show the status of the x-linked ring and show the next state after applying R-rule and S-rule to the x-linked ring, respectively. Then, the program will start to solve the x-linked ring and show the detailed operations step by step. Finally, the program will show the goodbye message. Note that you need to use recursion to complete the program and cannot use any loop (i.e., **for…**, **while…** or **do...while)** in the entire program. Otherwise, you will get 0 point.

Input

Positive integer X which represents the number of the rings with range [0 ~2^{31}-1].
X integers which represent state of each ring with range [0~1]

Output

The program will show a welcome message and ask the user to enter an integer *X*. Then, the program will ask the user to enter the current state of each ring form inside to the outside. Later, the program will show the current state of the x-linked ring. Then, the program will show the state after applying R-rule and S-rule to the current state of the x-linked ring, respectively. After that, the program will start to solve the x-linked ring problem and show the steps one by one. Then, the program will tell the user how many steps it takes. Note that all the output information related to rings should be shown with its ordinal information (i.e., 1st, 2nd, …).

Sample

Note: ***Italic*** shows the input from keyboard.

```
Welcome to play X-Linked Ring!
How many X-Linked Ring do you want to solve?
4

What the 4-Linked Ring look like?
Please enter the rings state from inside to outside.
If the ring is on the sword, please input 1. Otherwise, please enter 0.
What the state of 1st rings?
1
What the state of 2nd rings?
1
What the state of 3rd rings?
1
What the state of 4th rings?
1

The rings state of 4-Linked Ring is: 1111
If run R-rule once, the rings state of 4-Linked Ring is: 1110
If run S-rule once, the rings state of 4-Linked Ring is: 1101

That start to solve the 4-Linked Ring.
Start with S-rule !!
!! Turn the 3rd ring down !!
The rings state of 4-Linked Ring is: 1101
!! Turn the 4th ring down !!
The rings state of 4-Linked Ring is: 1100
!! Turn the 1st ring down !!
The rings state of 4-Linked Ring is: 0100
!! Turn the 4th ring on !!
The rings state of 4-Linked Ring is: 0101
!! Turn the 3rd ring on !!
The rings state of 4-Linked Ring is: 0111
!! Turn the 4th ring down !!
The rings state of 4-Linked Ring is: 0110
!! Turn the 2nd ring down !!
The rings state of 4-Linked Ring is: 0010
!! Turn the 4th ring on !!
The rings state of 4-Linked Ring is: 0011
!! Turn the 3rd ring down !!
The rings state of 4-Linked Ring is: 0001
!! Turn the 4th ring down !!
The rings state of 4-Linked Ring is: 0000

The 4-Linked Ring is solved in 10 step.
Thanks for using!! Goodbye ~
```

Bonus

In the bonus version, you need to use **while… / do…while…** to complete the program (i.e., iterative version.) Note that you can not use any **for…** in this program. Besides, you are asked to split your programs with the following functions:

- Inputrings(…){};
  The function will ask user to enter the current state of each ring, and stored the input information.
- Outputrings(…){};
  The function will show the current state of each ring.
- R_rule (…){};
  The function will provide R-rule to operate the rings.
- S_rule(…){};
  The function will provide S-rule to operate the rings.
- SolveRings(…){};
  The function will show how to solve the x-linked ring problem step by step.

The output of the bonus version should be identical to the recursion version. Therefore, the sample input/output of the bonus version is omitted. You need to separate the bonus version with your original version in two files. If you do not follow the rules, you can not get any point. Please follow the rules. Thank you!

Submission Requirement

You have to submit a source code file (not the entire project) named as StudID_PA2.cpp (ex: 9862534_PA2.cpp) and a report named StudID_Name_PA2_report.pdf (ex: 9862534_陳聿廣_PA2_report.pdf). If you implement a bonus version, please separate it from the original version. Name your source code file of the bonus version as StudID_PA2_bonus.cpp (ex: 9862534_PA2_bonus.cpp) and upload to ee-class with the original version. Note that the only acceptable report file format is .pdf, no .doc/.docx or other files are acceptable. **BE SURE to follow the naming rule mentioned above. Otherwise, your program will be not graded.**

We don't restrict the report format and length. In your report, you must at least describe:

1. How to compile and execute your program; (You can use screenshot to explain)
2. The completion of the assignment; (If you complete all requirements, just specify all)
3. The hardness of this assignment and how you overcome it;
4. Bonus function(s) you implement if any
5. Any suggestions about this programming assignment?

You can also put anything related to the PA in your report, such as pseudocode, control flow diagram, programming developing thought, etc. Your program will be judged with our On-line Judge (OJ) with the environment as follows:

OS：Ubuntu 20.04 LTS

OJ version：Hydro OJ v2.33.13

Compiler：GNU GCC 9.3.0

C++ version：C++ 11

Note that we will provide limited public testcases and your program will be judged by both public cases and hidden cases. The on-line judge information will be announced later.

Grading

The grading is as follows:

(1) Correctness of your code: 50%

(2) Readability of your code: 10%

(3) The report: 10%

(4) Demo session: 30%

(5) Bonus (at most): 10%

Please submit your assignment on time. Otherwise, the penalty rule will apply:

- Within 72hrs delay: 20% off
- More than 3 days: 0 point

Be sure to attend a demo session (the time will be announced later). If you have questions, please E-mail to both me (andyygchen@ee.ncu.edu.tw) and TA 林俐秀 (sophia30516@gmail.com)

Reference

[1] Chinese Nine Linked Rings Puzzle

[2] 郭君逸,「九連環與格雷碼」數學傳播 38 卷 3 期, pp. 13-24

[3] 九連環模擬器