Lab 3: Tower of Hanoi

In Lab 3, you are required to design and implement a game: Tower of Hanoi.

0 The puzzle

The Tower of Hanoi is a famous mathematical puzzle. It consists of three rods and a number of disks of different sizes. Initially, all disks are in the first rod in ascending order of size (i.e., the smallest is at the top). The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upmost disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

The classical way to solve the puzzle is recursion. The algorithm could be described as the following pseudocode:

```
function hanoi(n, A, B, C) {  // move n disks from rod A to rod B, use rod C as a buffer
     hanoi(n - 1, A, C, B);
     move(n, A, B);  // move the nth disk from rod A to rod B
     haoni(n - 1, C, B, A);
}
```

We also devise a **Enhanced Tower of Hanoi** puzzle. In enhaced tower of Hanoi, the goal is to let the biggest `a` disks be in the first rod, the second biggest `b` disks in the second rod and the remaining `c` disks in the thrid rod. All `a+b+c` disks are in the first rod at the beginning.

You should solve the enhaced tower of hanoi with the following algorithm:

```
function hanoi_enhanced(a, b, c, A, B, C) {
    hanoi(b + c, A, B, C); // move b+c disks from rod A to rod B
    hanoi(c, B, C, A); // move c disks from rod B to C
}
```

1 Lab Description

Here are some concepts that you need to know first.

- Canvas: The UI of the game.
- Rod: There are 3 rods (numbered 1, 2 3) in the canvas. Disks are pushed and popped among

them. The source rod is 1 and the target rod is 2. Rod 3 acts as a temporary rod.
- Disk: Each disk is of different sizes. In the beginning, they are all pushed into Rod 1 in ascending order of size.

You should implement the Hanoi game program with the following functionality:

1. **Intial Phase:** When the program starts, it firstly asks the player for the number of disks. The input should a integer in **[1, 5]**. If input is `Q` , quit the game. Other invalid input should be ignored.
2. **Normal Mode:** Then, print the canvas in the console, and the game starts. The program continuously asks for commands until the player wins. The canvas will be printed after each move. Command format is: `from to` (for example `2 3` moves the topmost disk from 2 to 3). Invalid input should be ignored.
3. **Auto Play Mode:** If the game receives command `0 0`. Auto-play mode will be activated. The game will automatically continue without player's commands. The canvas, as well as the auto-generated command, will be printed after each move.
4. **Enhanced Auto Play Mode** If the game receives command `-a -b -c`, where `a`, `b` and `c` are non-negative numbers and `a + b + c` equals total number of disks. Enhanced auto-play mode will be activiated. The game will automatically continue without player's commands like auto player mode, but the goal change to **Enhanced Tower of Hanoi**.

Requirements

- You have to implement the game using **Object-Oriented** programming. Hope the following materials will help you if you're not familiar with OOP.
  - Object Oriented Design
  - Object Oriented Programming in C++
- Containers in STL(`std::vector, std::stack, std::list, std::queue`) are **NOT** allowed to use. Please implement your own `Stack` or `Queue` if needed.

2 Guidance

2.1 Drawing the Canvas

We have offered you a file, `termio.h`. It might be helpful for you to deal with the canvas printing work.

- `CANVAS_WIDTH` and `CANVAS_HEIGHT`: the width and height of the canvas.
- `Clear()` helps you clear the screen. Using it before `Draw()` will improve the user experience.
- `Draw()` helps you draw the buffer to console.
- `ResetBuffer()` fills the buffer with spaces.

2.2 Auto Play Mode

During the game, players may enter auto-play mode in any state. After entering into auto-play mode, you need to do two things.

- Restore to the beginning state step by step (simple idea: use a stack to log the player's commands and undo these commands when entering auto-play mode).
- Auto-play the game from the beginning state.

2.3 Enhanced Auto Play Mode

Enhanced auto play mode is the same with Auto play mode. The only difference is that the goal is to let there be `a`, `b`, `c` disks in rod 1, 2 and 3.

2.3 Output format

- For the sake of simplicity, the disk is represented by `*` .
    - The number of `*` that you should draw is calculated by `2* disk_size + 1`. For example, if there are five disks in rod1, then the number of `*` is `3, 5, 7, 9, 11`. The middle `*` is vertically aligned with the rod .
    - You should draw disks from the bottom of the rod. The two examples below show the canvas when there are 5, 3 disks at the beginning.
    - The position of the rods should remain unchanged regardless of number of disks.
    - We have defined `CANVAS_WIDTH` and `CANVAS_HEIGHT` for you, you should not change the value.

```
     |             |             |
    ***            |             |
     |             |             |
   *****           |             |
     |             |             |
  *******          |             |
     |             |             |
 *********         |             |
     |             |             |
***********        |             |
-----|--------------|--------------|-----
```

```
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
    ***             |              |
     |              |              |
   *****            |              |
     |              |              |
  *******           |              |
-----|--------------|--------------|-----  // this line should be unchanged
regardless of number of disks
```

- Upon wining the game (either from normal mode or auto-play mode), first print "Congratulations! You win!", and then start a new round of game.

  `std::cout << "Congratulations! You win!" << std::endl;`

  `std::cout << "How many disks do you want? (1 ~ 5)" << std::endl;`

- About other output formats, please see the example below.

3 Example

3.1 Initial Phase

```
How many disks do you want? (1 ~ 5)
2
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
    ***             |              |
     |              |              |
   *****            |              |
-----|--------------|--------------|-----
Move a disk. Format: x y
```

After telling the program about the number of disks, we come to the normal mode.

3.2 Normal Mode

```
1 3
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
   *****            |             ***
-----|--------------|--------------|-----
Move a disk. Format: x y
1 3
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
   *****            |             ***
-----|--------------|--------------|-----
Move a disk. Format: x y
```

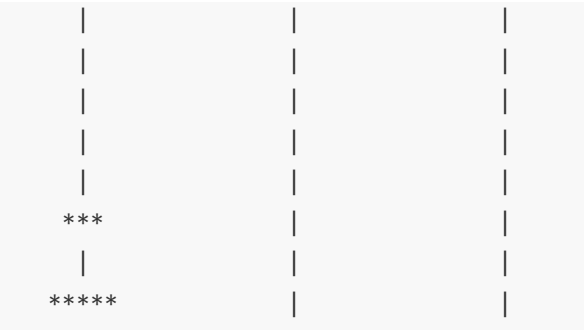
As we can see, when we input `1 3` for the first time, the top disk of Rod 1 is popped and pushed to Rod 3.

Then we give an illegal command `1 3` (since it violate the rule: No larger disk may be placed on top of a smaller disk.), it will be ignored, so the previous canvas is printed.

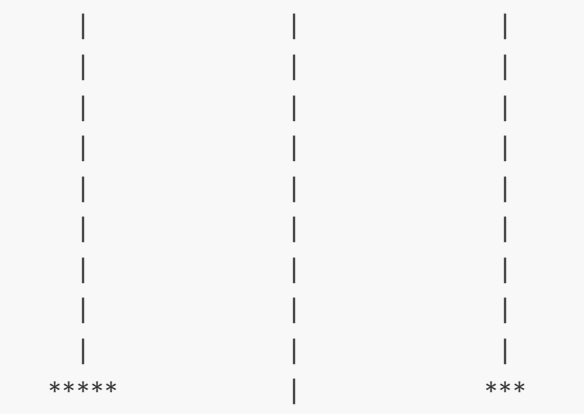
3.3 Auto Play Mode

Then we tried the auto-play mode by giving the command `0 0`. As we can see, the program executes automatically from the begining state until the player wins.

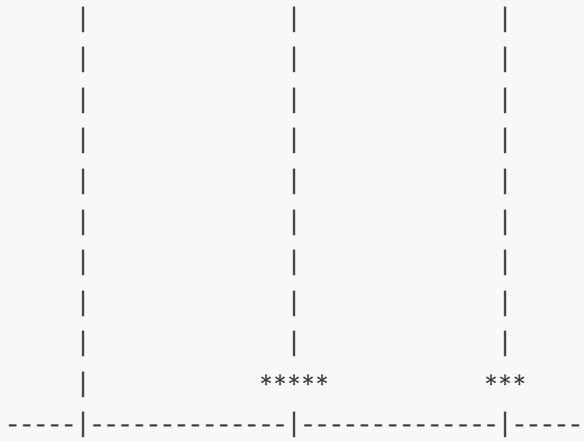
```
0 0
Auto moving:3->1  // restore the the beginning state first
     |              |              |
     |              |              |
```



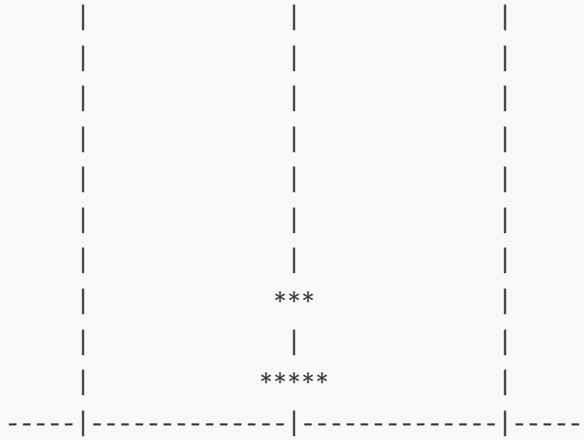
Auto moving:1->3



Auto moving:1->2



Auto moving:3->2



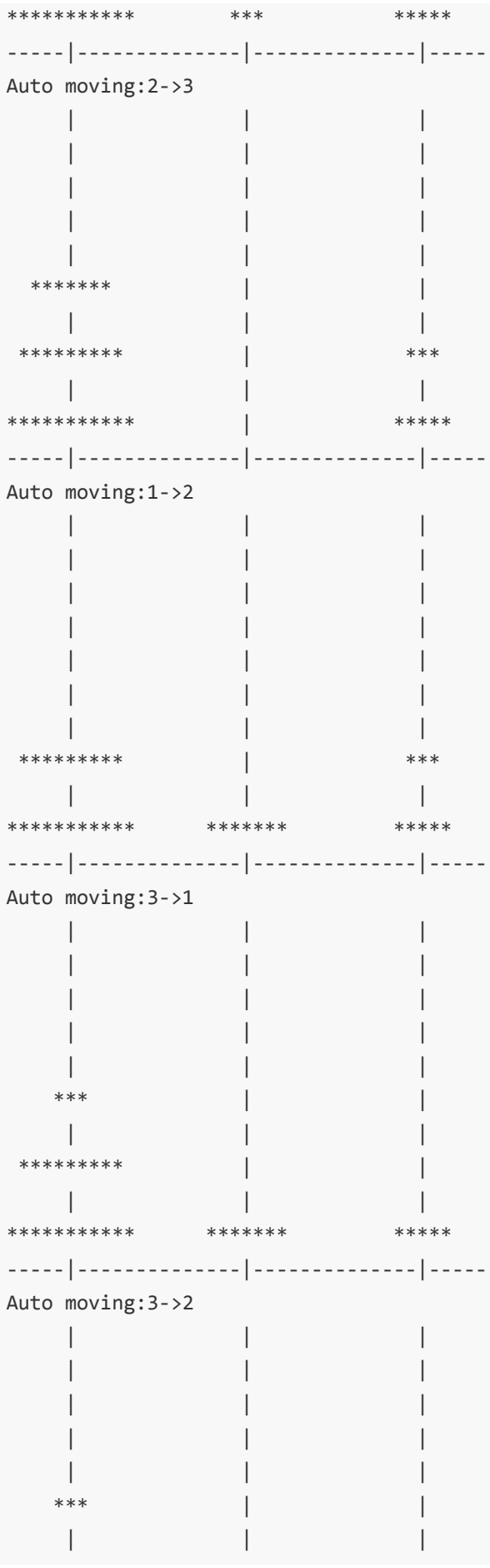
Congratulations! You win!

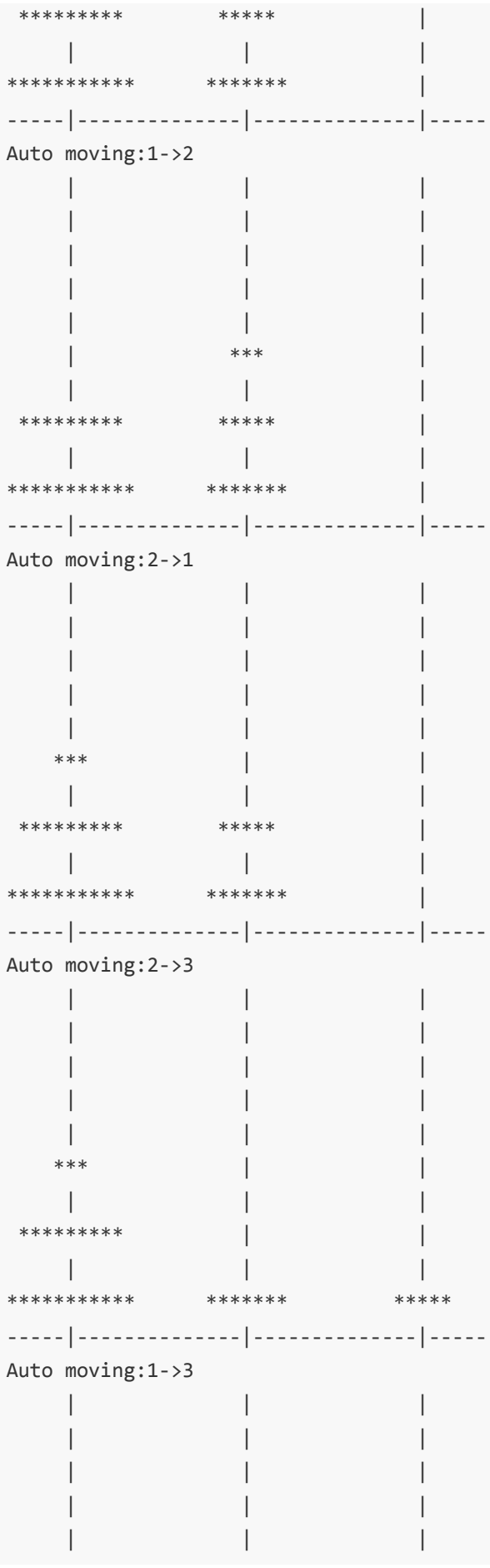
```
How many disks do you want? (1 ~ 5)
Q   // quit the game now
```

3.4 Enhanced Auto Play Mode

Let's restart the game and try the enhanced auto play mode with command `-2 -1 -2`. As we can see, the program executes automatically from the begining state until the player wins.

```
How many disks do you want? (1 ~ 5)
5
     |              |              |
    ***             |              |
     |              |              |
   *****            |              |
     |              |              |
  *******           |              |
     |              |              |
 *********          |              |
     |              |              |
***********         |              |
-----|--------------|--------------|-----
Move a disk. Format: x y
-2 -1 -2
Auto moving:1->2
     |              |              |
     |              |              |
     |              |              |
   *****            |              |
     |              |              |
  *******           |              |
     |              |              |
 *********          |              |
     |              |              |
***********        ***             |
-----|--------------|--------------|-----
Auto moving:1->3
     |              |              |
     |              |              |
     |              |              |
     |              |              |
     |              |              |
  *******           |              |
     |              |              |
 *********          |              |
     |              |              |
```


****
***
*****
-----|----------|----------|-----
Auto moving:2->3
****
****
***
****
*****
-----|----------|----------|-----
Auto moving:1->2
****
***
****
****
*****
-----|----------|----------|-----
Auto moving:3->1
***
****
****
****
*****
-----|----------|----------|-----
Auto moving:3->2
***


Auto moving:1->2
Auto moving:2->1
Auto moving:2->3
Auto moving:1->3

```
     |             |             |
     |             |             |
 *********         |            ***
     |             |             |
***********      *******        *****
-----|-------------|-------------|-----
Congratulations! You win!
How many disks do you want? (1 ~ 5)
Q   // quit the game now
```

4 Submission

Compress your source code into a 7z file, and rename it to `lab3-XXX.7z` where `XXX` is your student id. The `lab3-XXX.7z` file should contain a folder named `lab3`. Upload `lab3-XXX.7z` to canvas.

The 7z file should have a folder structure like this:

```
lab3-XXX.7z
   |--- lab3
          |--- xxx.h
          |--- xxx.cc
          |--- xxx.pro
          |--- Makefile
          |--- ...
```