Advanced Programming

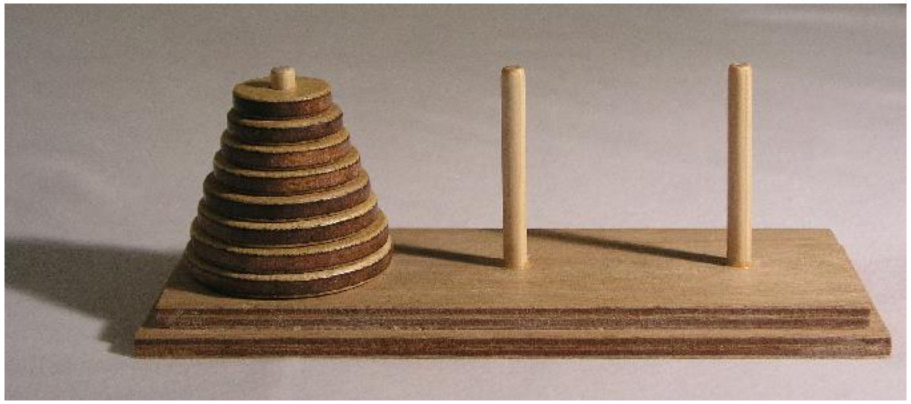
Programming Assignment #1

Kang Hoon Lee

Kwangwoon University

Tower of Hanoi: A Mathematical Puzzle

- **Goal: Move the entire stack of disks from the first rod to the last one while obeying the following rules:**
  - Only one disk can be moved at a time.
  - Each move consists of taking the upper disk from one of the stack and placing it on top of another stack or on an empty rod.
  - No larger disk may be placed on top of a smaller disk.



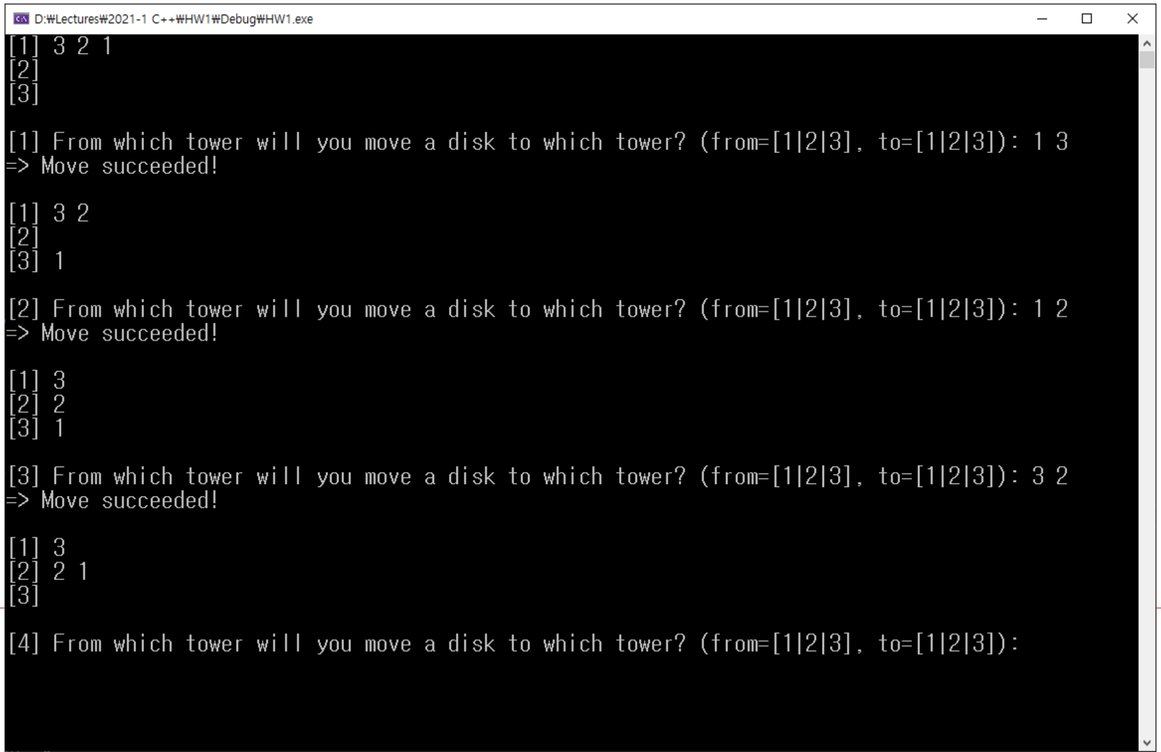
※ With n disks, the minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$

https://en.wikipedia.org/wiki/Tower_of_Hanoi

http://towersofhanoi.info/Default.aspx

Interactive App for Solving Tower of Hanoi

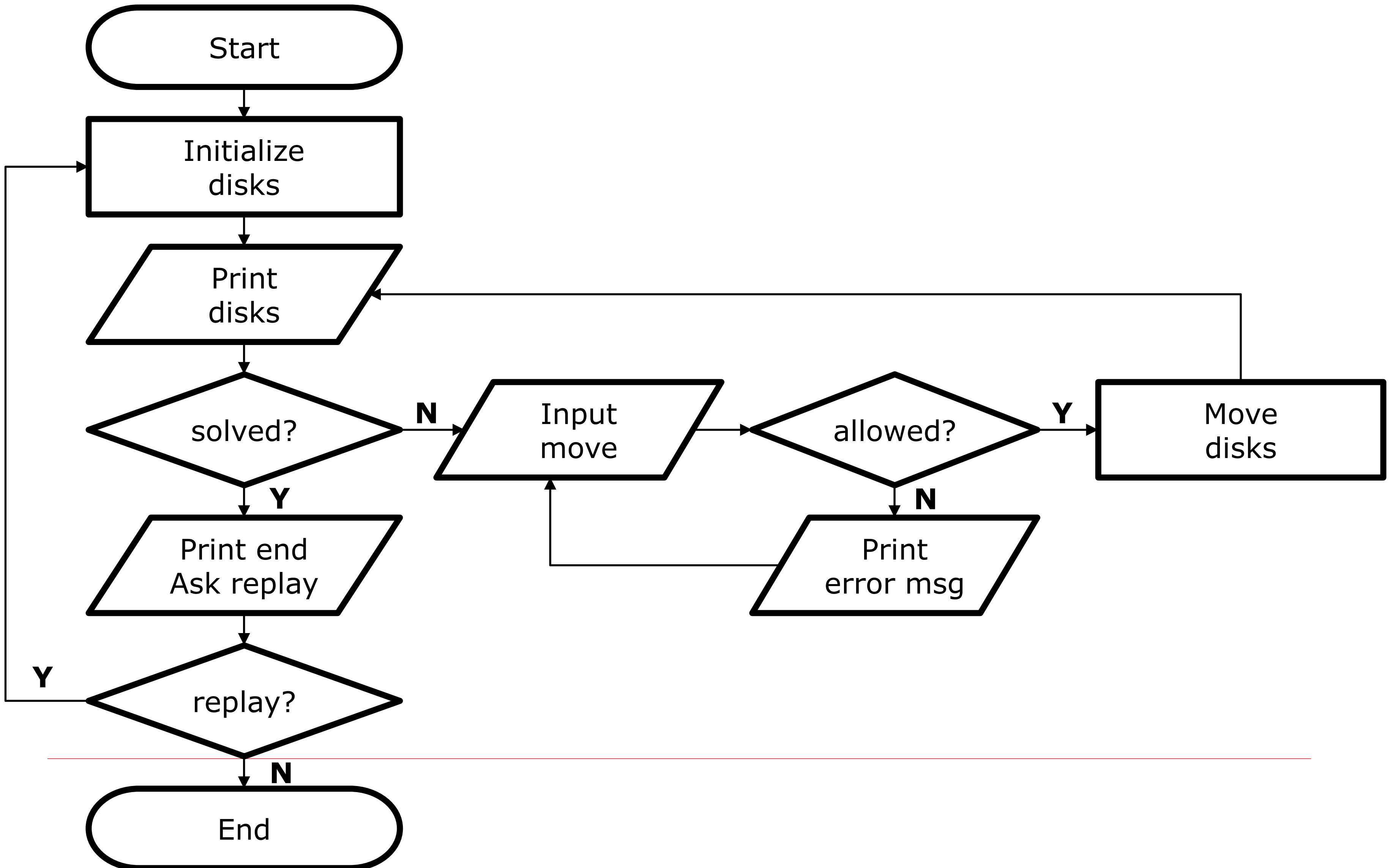
- **Objective**
  - Implement a text-based application of solving Tower of Hanoi puzzle in a console window by using C++
  - An example screenshot



D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

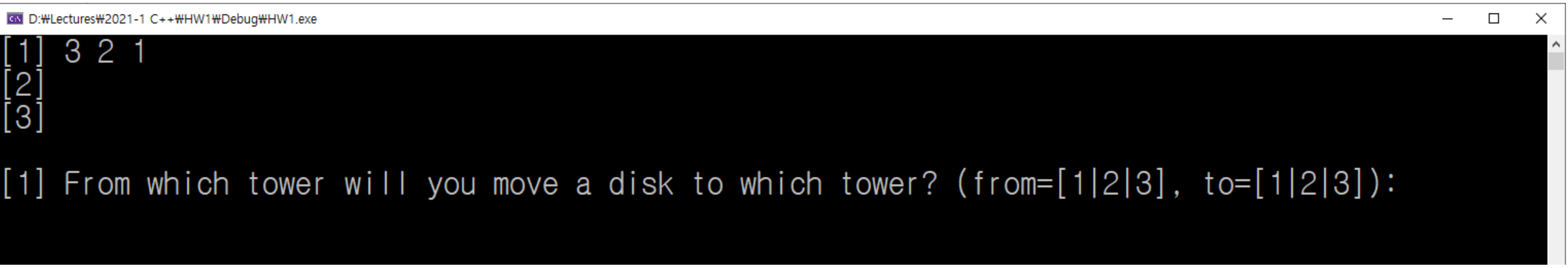
```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1] 3 2
[2]
[3] 1

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 2
=> Move succeeded!

[1] 3
[2] 2
[3] 1

[3] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 3 2
=> Move succeeded!

[1] 3
[2] 2 1
[3]

[4] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Flow Chart


Start
Initialize disks
Print disks
solved?
N
Input move
allowed?
Y
Move disks
Y
N
Print end Ask replay
Print error msg
replay?
Y
N
End

Example Results

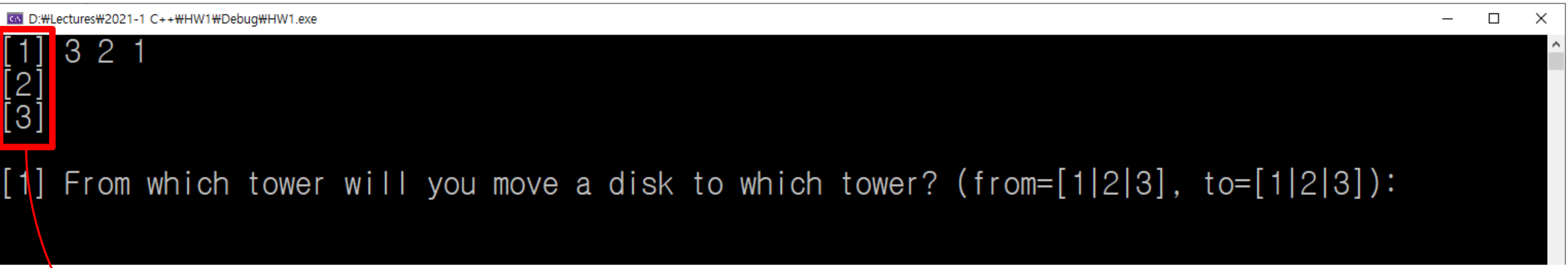
- **Initial status**
  - All disks are stacked on the first rod in a decreasing order of sizes



```
D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Example Results

- **Initial status**
  - All disks are stacked on the first rod in a decreasing order of sizes

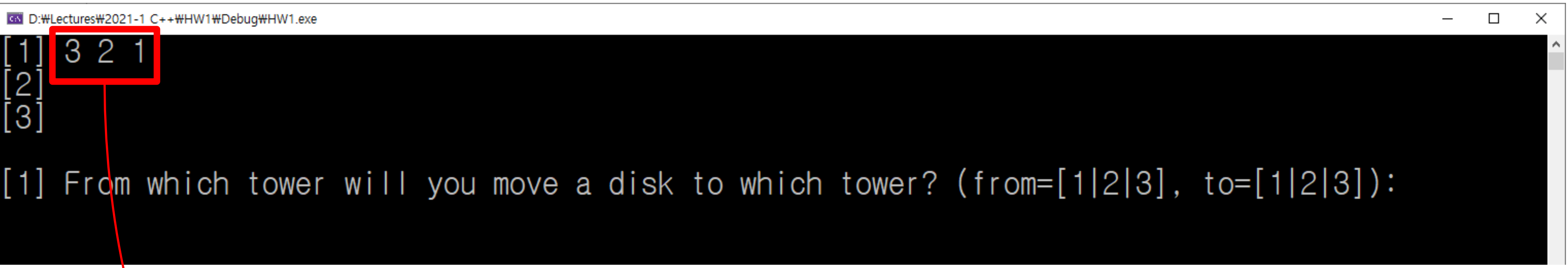


```
D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Indices of rods (starts from **1**, not **0**)

Example Results

- **Initial status**
  - All disks are stacked on the first rod in a decreasing order of sizes



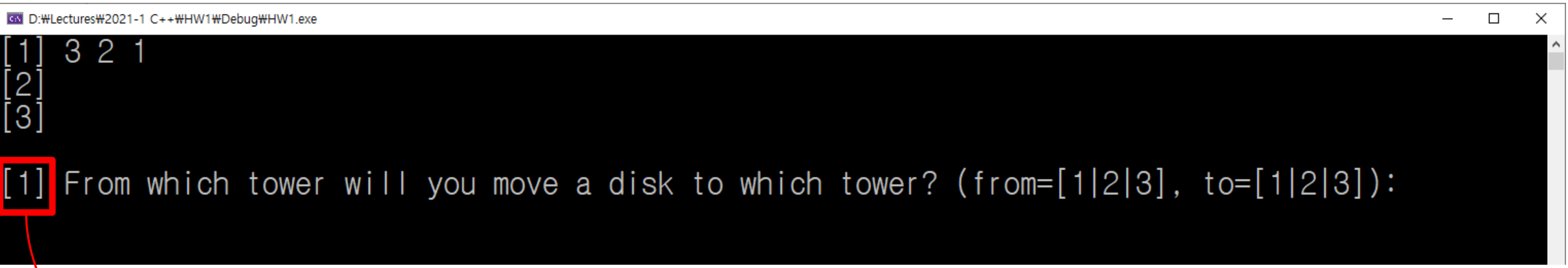
D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Sizes of disks stacked on a rod (from bottom to top)

Example Results

- Initial status
  - All disks are stacked on the first rod in a decreasing order of sizes

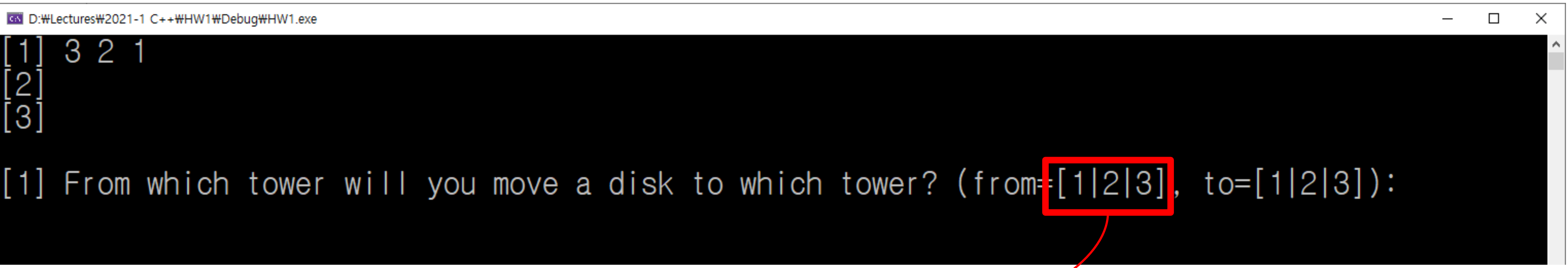


```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Number of moves (starts from **1** again, not **0**)

Example Results

- **Initial status**
  - All disks are stacked on the first rod in a decreasing order of sizes



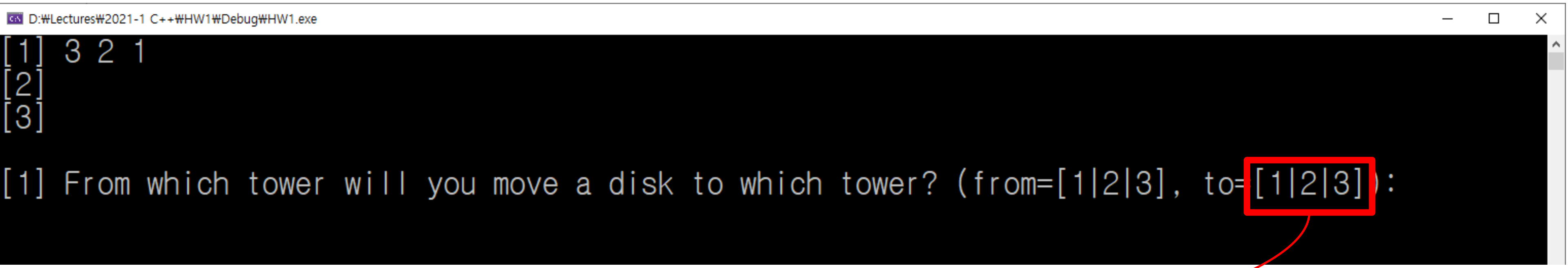
D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

The allowed indices of the rod **from** which a disk will be moved

Example Results

- **Initial status**
  - All disks are stacked on the first rod in a decreasing order of sizes



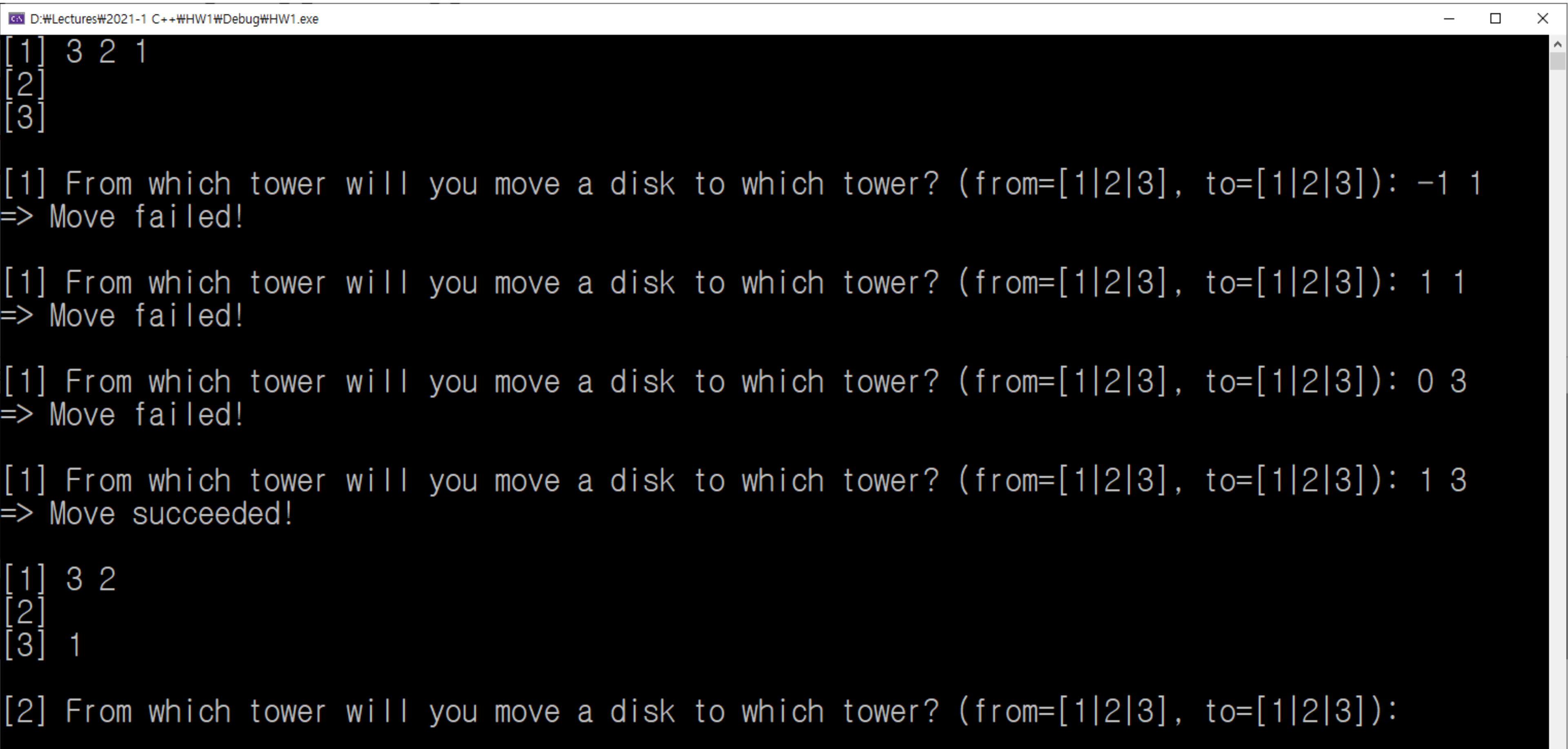
D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

The allowed indices of the rod **to** which a disk will be moved

Example Results

- **Receive the next movement input from the user**
  - The user needs to give the indices of the rods from and to which a disk will be moved within the allowed range (1~3)

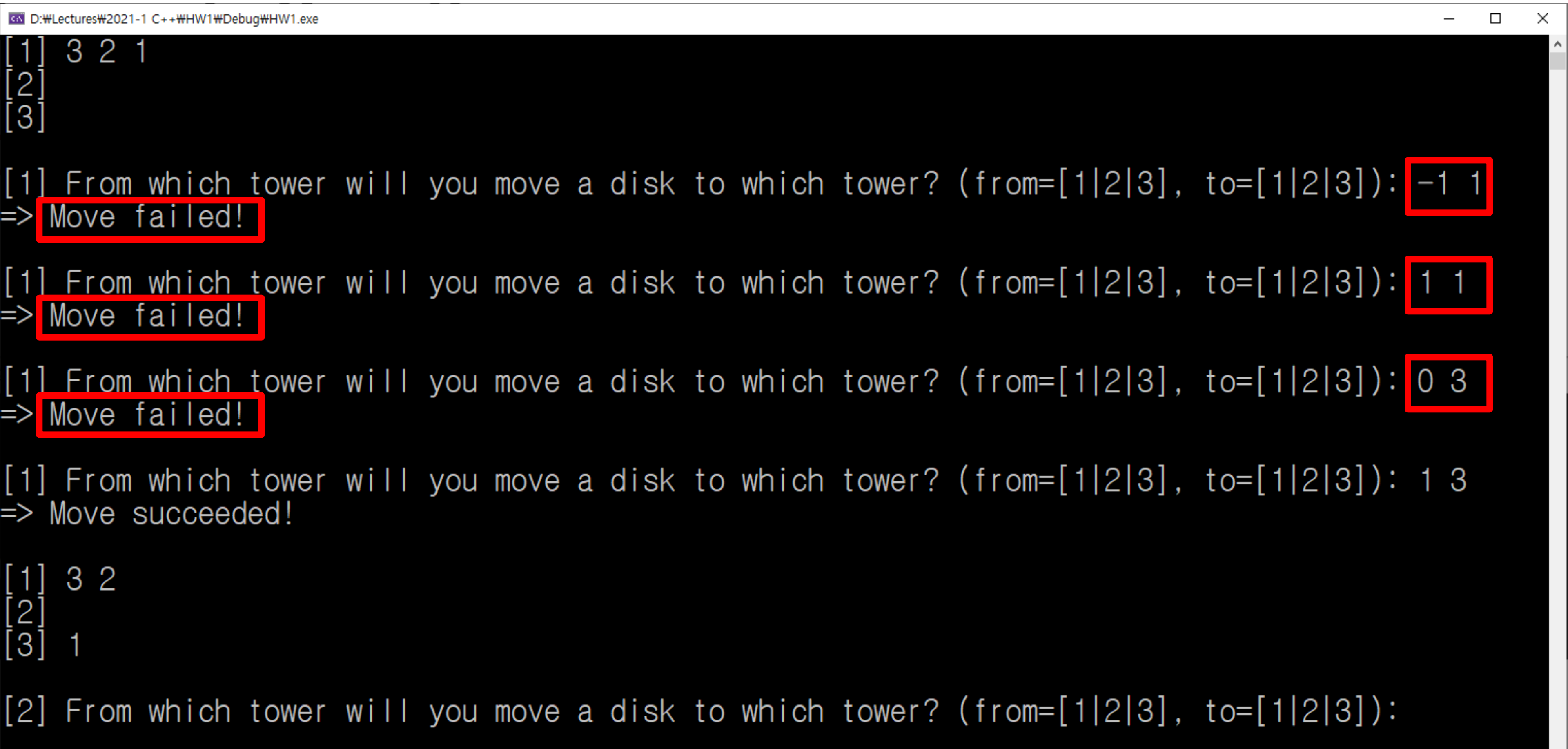


D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): -1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 0 3
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1] 3 2
[2]
[3] 1

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Example Results

- **Receive the next movement input from the user**
  - The user needs to give the indices of the rods from and to which a disk will be moved within the allowed range (1~3)

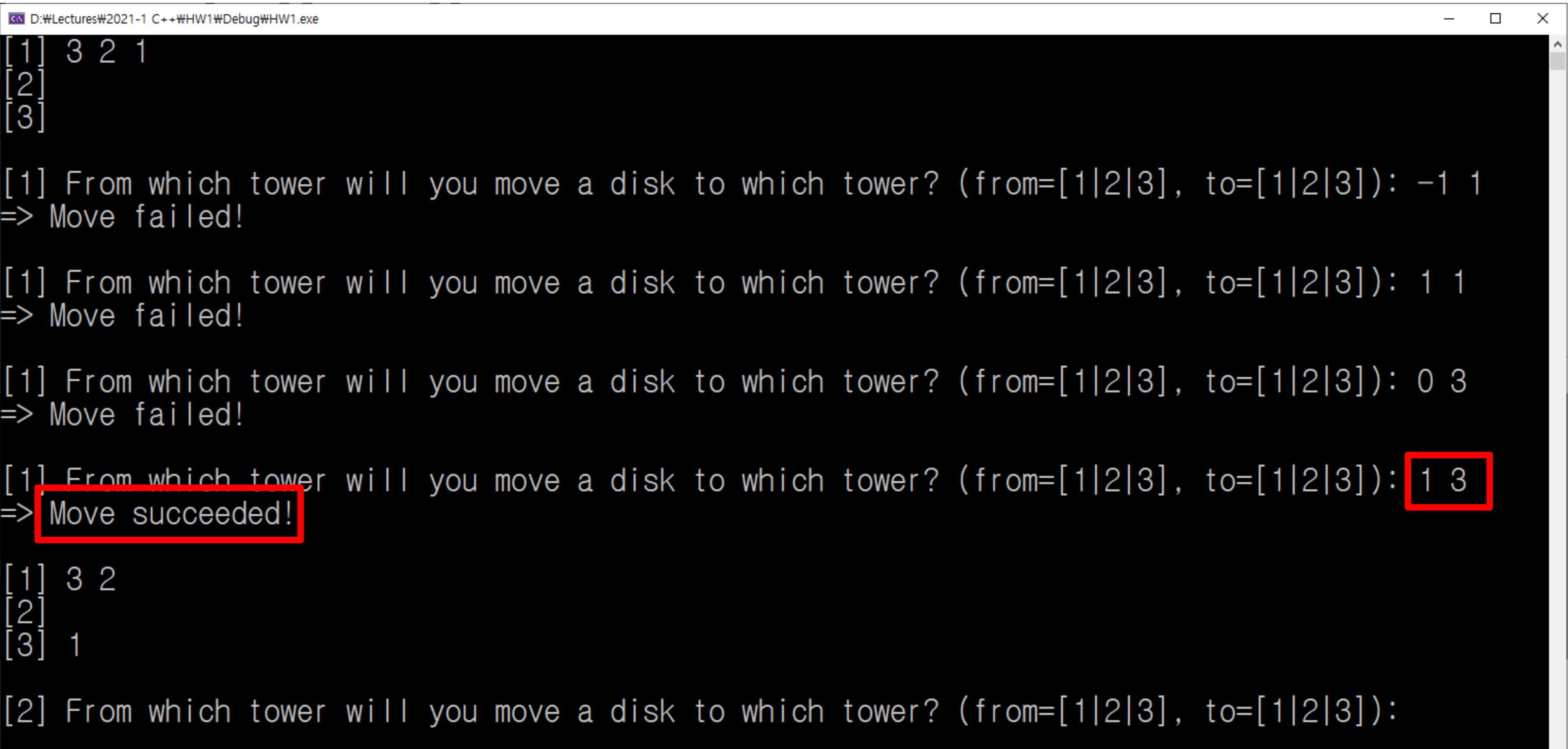


D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): -1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 0 3
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1] 3 2
[2]
[3] 1

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Example Results

- **Receive the next movement input from the user**
  - The user needs to give the indices of the rods from and to which a disk will be moved within the allowed range (1~3)

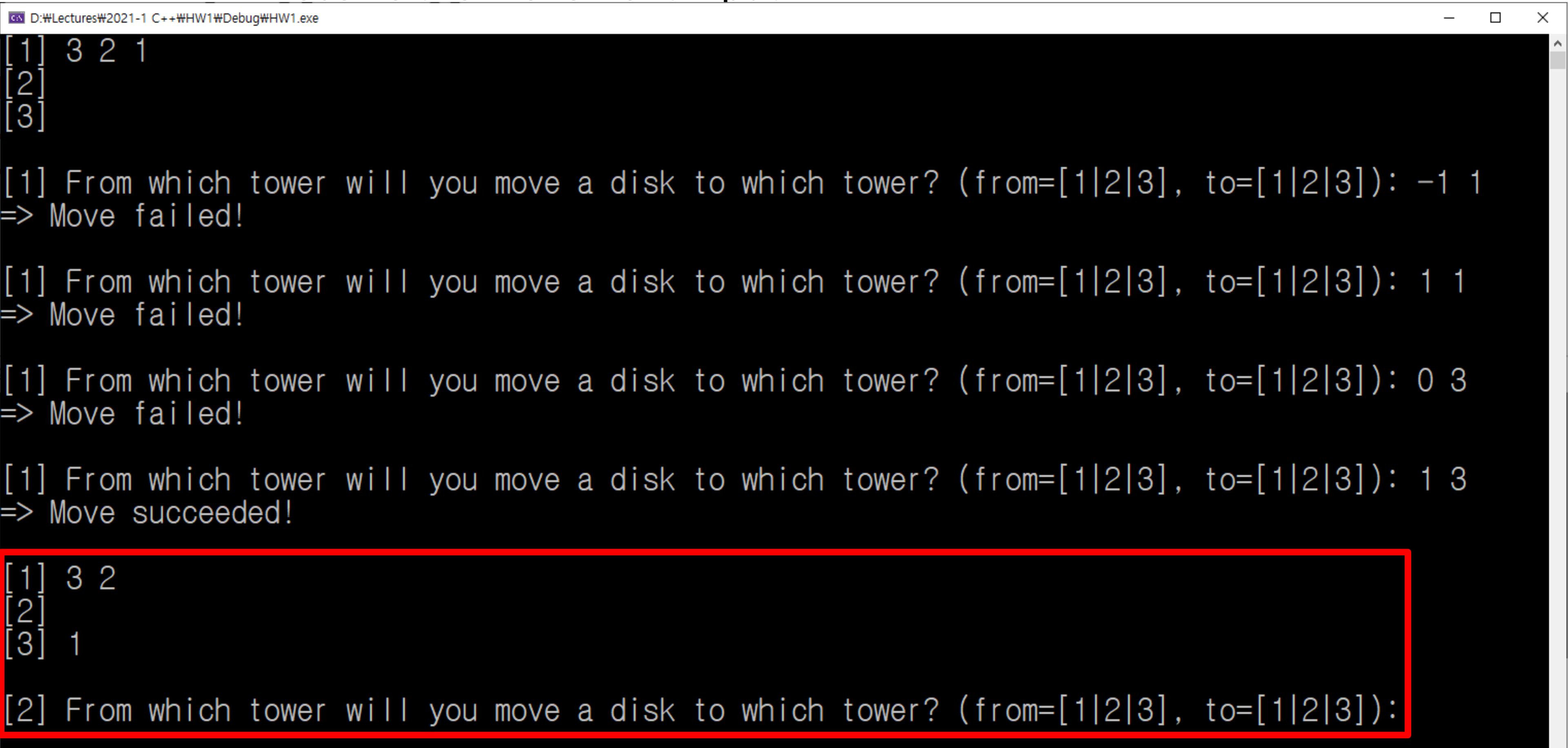


D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): -1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 0 3
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1] 3 2
[2]
[3] 1

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Example Results

- **Repeat the same process till the puzzle is solved**
  - Print the updated status
  - Receive the movement input

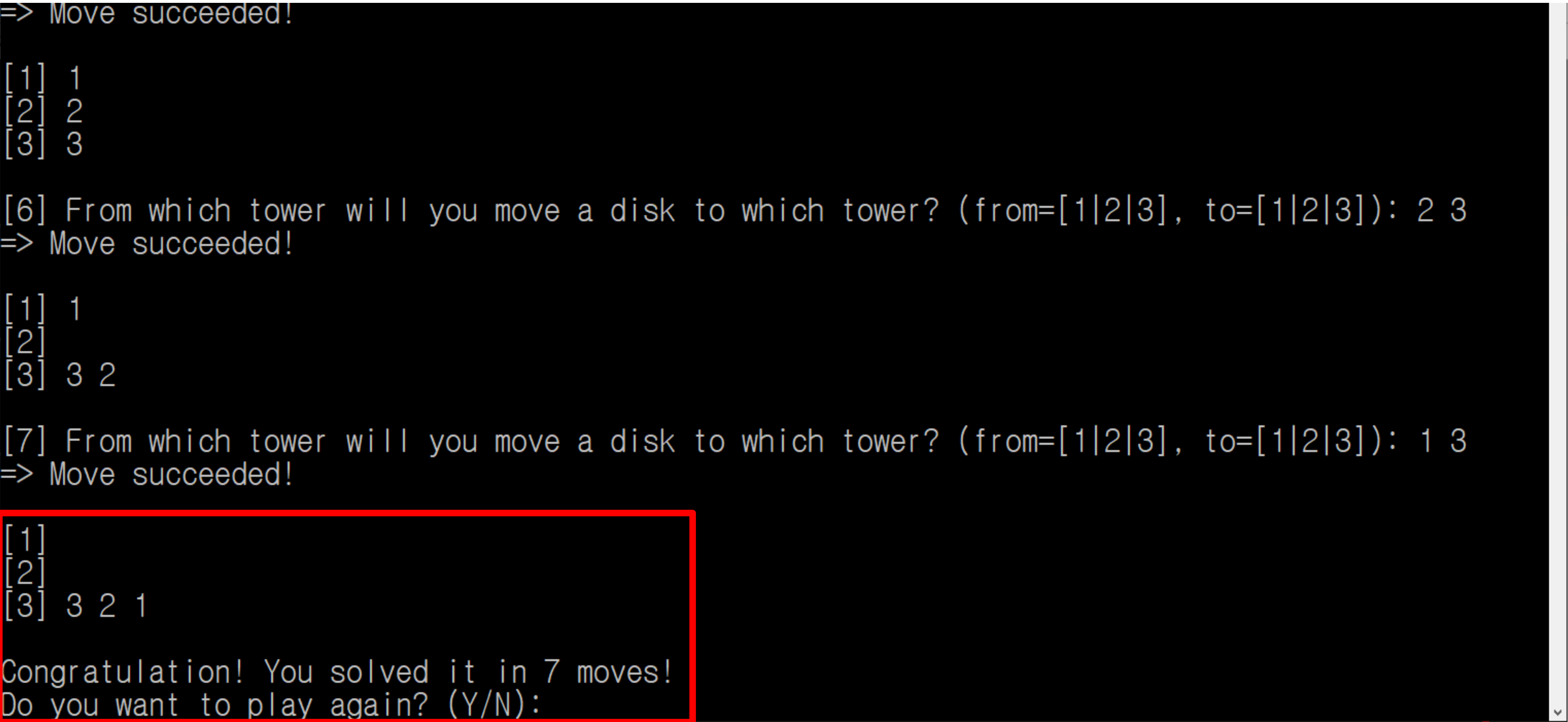


D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
[1] 3 2 1
[2]
[3]

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): -1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 1
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 0 3
=> Move failed!

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1] 3 2
[2]
[3] 1

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```

Example Results

- Terminate the current session if the puzzle is solved
  - Print the congratulation message
  - Allow the user to solve the puzzle again from the initial status



```
=> Move succeeded!

[1] 1
[2] 2
[3] 3

[6] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 2 3
=> Move succeeded!

[1] 1
[2]
[3] 3 2

[7] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
=> Move succeeded!

[1]
[2]
[3] 3 2 1

Congratulation! You solved it in 7 moves!
Do you want to play again? (Y/N):
```

Submission

- **Code**
  - Include only the "**`std_lib_facilities.h`**"
  - Use **`cin`** and **`cout`** for input and output
    - Don't use printf and scanf
  - Use **`vector`** to store disks on each rod
    - Don't use arrays of anything
    - Use `push_back()`, `pop_back()`, `size()`, `clear()`, and [ ] operator
    - *Hint.* You can even use a vector of vectors to store disks on all rods
      - (e.g. `vector<vector<int>> towers;`)
  - Define *at least* **3 functions** in addition to **`main()`**
    - e.g. `PrintTowers(), IsMoveAllowed(), MoveDisk(), etc.`
  - Do **NOT** declare any global variables
    - Use only local variables and function arguments
  - Make the code readable
    - Meaningful names, indentation, comments, etc.
  - Grading will be done with **Visual Studio Community 2019**

Submission

- **Report**
  - Title page
    - Course title, submission date, affiliation, student ID, full name
  - For each requirement, explain how you fulfilled it
    - Do not just dump the entire code at once
    - It's okay to copy snippets of your code to complement written description
  - For each additional feature, if exist, explain what it is and how you implemented it
    - e.g. drawing the status of puzzle more intuitively, solving the puzzle automatically, allowing to modify # of disks, # of rods, etc.
  - Demonstrate the correctness of your code
    - How correctly control the overall flow as specified in the flow chart
    - How correctly respond to valid and erroneous inputs
    - How correctly determine if the puzzle is solved or not
    - Capture and attach a screen shot for each example case
  - Conclude with some comments on your work
    - Key challenges you have successfully tackled
    - Limitations you hope to address in the future

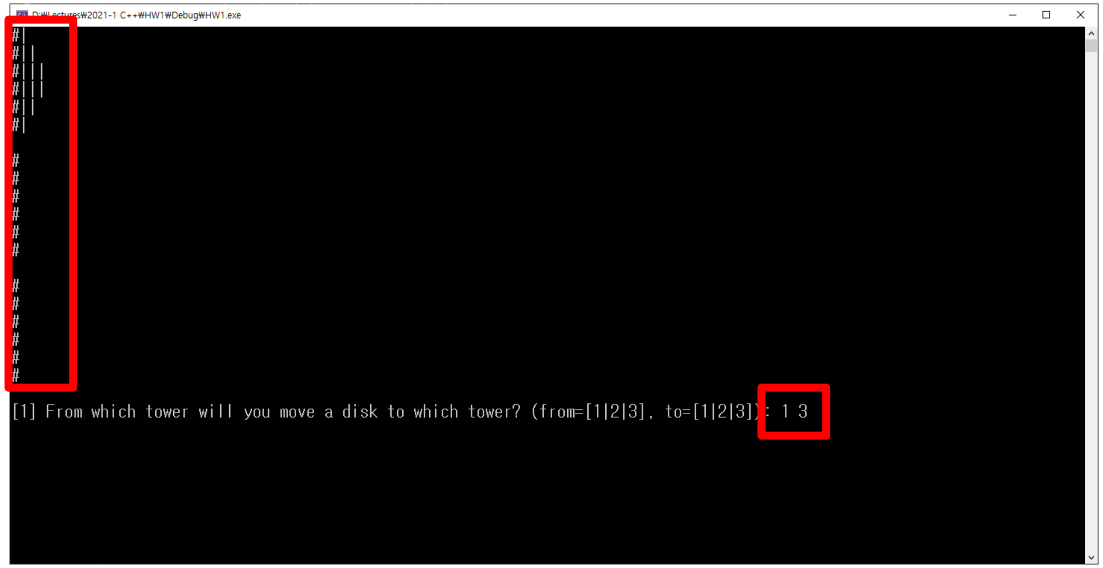
Submission

- **Compress your code and report into a single *.zip file**
  - Code
    - The entire project folder including *.sln, *.cpp, *.h, etc.
    - The grader should be able to open the *.sln and build/run the project immediately without any problems
  - Report
    - A single *.pdf file
    - You should convert your word format (*.hwp, *.doc, *.docx) to PDF format (*.pdf) before zipping
  - Name your zip file as your student ID
    - ex) 2012726055.zip

- **Upload to homework assignment menu in KLAS**

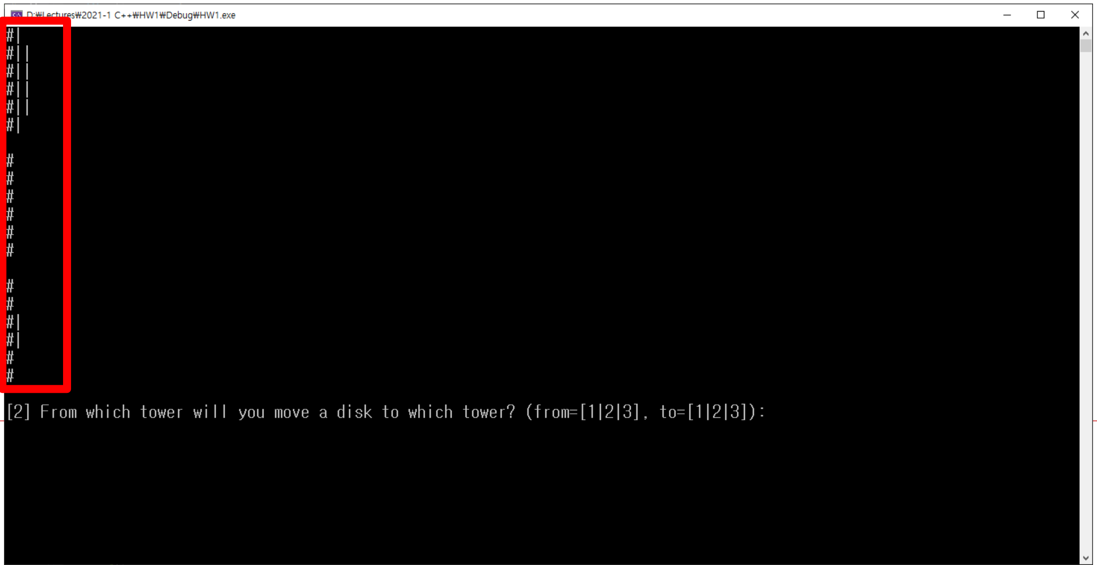
- **Due at 4/10 (Sat), 11:59 PM**

An Example of Additional Features



D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
#|
#||
#|||
#|||
#||
#|

#
#
#
#
#
#

#
#
#
#
#
#

[1] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]): 1 3
```



D:₩Lectures₩2021-1 C++₩HW1₩Debug₩HW1.exe

```
#|
#||
#||
#||
#||
#|

#
#
#
#
#
#

#
#
#|
#|
#
#

[2] From which tower will you move a disk to which tower? (from=[1|2|3], to=[1|2|3]):
```