Writer's Name: Mubasshir Al Shahriar (Nihal)

Assignment: Non-recursive Towers of Hanoi

The Tower of Hanoi is a famous mathematical game which was invented by a French mathematician named Édouard Lucas in 1883. On this puzzle-game the players have three rods/sticks and a variable number of disks of different size/diameters. The condition of this puzzle is to arrange those disks on only one rod in decreasing order (depending on the diameter of disks). When you solve the puzzle, you should find a conical shape. In a nutshell, the conditions to maintain during the gameplay are:

i. Disks must be in decreasing order, meaning the smallest disk should be on the top and the largest at the bottom,

ii. You cannot put a larger disk on top of a smaller one, not even temporarily.

To better visualize it or play the game, you can play it online on:
https://www.mathsisfun.com/games/towerofhanoi.html

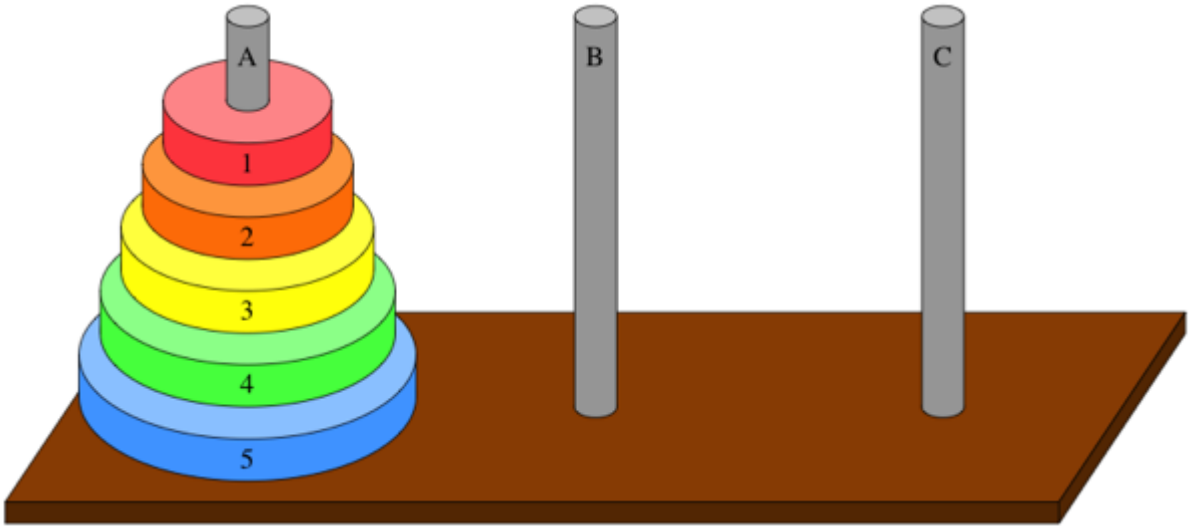


Image Credit: Khan Academy

There is a legend about an old Indian temple which contains a large room with three time-worn posts in it which is surrounded by 64 golden disks. The priests of Brahma, acting out the command of an ancient prophecy, have been moving these disks, in accordance with the rules of the puzzle which I described up in this page. The legend tells, when the last move of the disk will be completed, that day will be the dooms day and the universe will end. This puzzle is also known as the 'Tower of Brahma puzzle.' However, it is not evident whether mathematician Lucas himself invented this legend or he was inspired by this famous Old-Indian legend. But it will be evident for you soon how many moves you will need to solve the puzzle-game with n(variable) number of disks if you run the cpp program I coded and provided with this project. Then you can also know the predicted dooms daytime according to the legend. By using my algorithm and c++ code, I found that if the legend were true and at the same time if the priests were able to move all those 64 golden disks at a rate of 1 per second, and do that with least possible number of moves, like the algorithm I used for this project, then it would take them ((2^64) – 1) seconds which is around 585.44 billion years. But don't worry about that as this is just a very old legend and also the universe is currently only 13.7 billion years old approximately with an uncertainty of maximum 200 million years according to the latest research of the scientists of NASA .

Short Report/Summary:

- We have total "n" number of rings in tower A, and we have to move all of them to tower B in such manner that no larger ring is put over a smaller ring and we can move only the top ring from one tower.

- **Basic algorithm to solve this problem is: The tower from where we'd move a ring in a move we can call "from tower." The tower which will receive that ring, we can call "to tower." The ring which will be moved can be marked as "candidate." We have to keep moving the rings from "from tower" to "to tower" either clockwise or counterclockwise. It will depend on the total number of rings (n). If the total number of rings is odd, we'd need to move the rings clockwise, if that's even, then we have to move the rings counterclockwise. For example, if we have a total of 3 rings, in the very first move, we'll move the ring 1 from the top of tower A to tower B (clockwise). So, tower B will be the "to tower" for the first move in this case. If we have 4 rings, we'll move that ring 1 to tower C (anticlockwise). So, tower C is the "to tower" for the first move in this case, as "n" is even. After the first move, we have to select next "from tower." The "to tower" of the last move can't be considered as new from tower in very next move because, according to the condition, we can't move a ring which was just moved. So, the new from tower will be from the rest 2 towers. To select that, we've to check which tower has the smallest ring. The tower with the smallest ring will be the new from tower and the top ring of it will be the new candidate. To select a new "to tower" we'd first check which tower has the priority. If "n" is odd, then clockwise suitable next (right) tower will be new to tower, if there is no smaller ring than the candidate ring in that tower. If that tower with first priority is not suitable to receive candidate, then the tower with second priority will be the "to tower." And if "n" is even, it will be the vice-versa. Thus, we'd keep the rings moving until we move all the rings to the tower B.**

- In our program, we have initialized and set the value of “firstpriority” and “secpriority” depending on whether “n” (total number of rings) is even or odd. If it is even, we set the “firstpriority” to 2, and “secpriority” to 1, so that, it moves the rings counterclockwise. Otherwise, we set the value vice-versa to move the rings clockwise when “n” is odd.

- We initialized the towers by initializing the vector “t” and using push_back() function depending on “n.” To put “n” rings in the tower A, we used a for loop and set the value of

i = n + 1 because, because vector can't be empty, so, to keep the existence of all three towers, we needed to put an extra ring to every tower (by setting it to "n + 1").

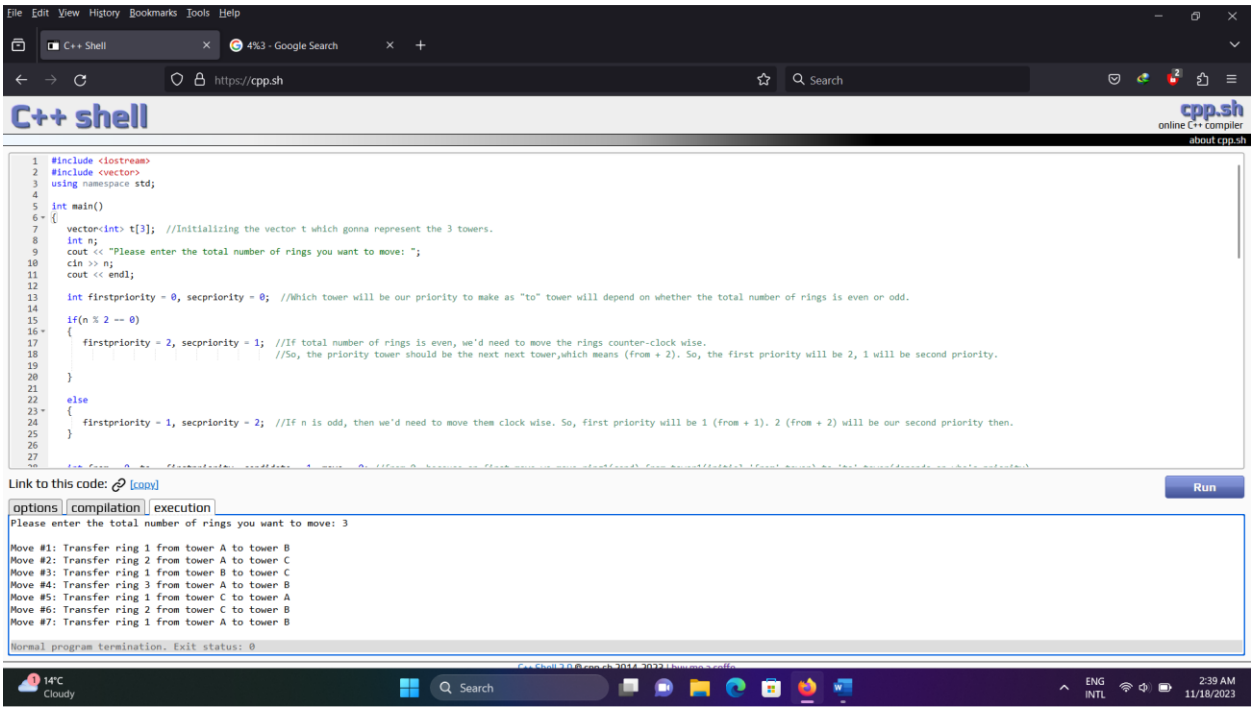
When we are moving a candidate ring from "from tower" to "to tower," in program we are actually copying the value of last element of that from tower to "to tower" using .back() and .push_back() function. After moving, as we copied to, so now we need to delete that element from the "from" tower. So, we used .pop_back() function to delete that candidate.

We used "%3" while selecting from and to tower because we've total 3 towers, and we have to move rings clockwise or counterclockwise staying within the loop of those 3 towers, we can't go beyond those 3 towers. "%3" enabled us to stay in that loop as (any number) % 3 will take us to a number within 0 to 2 (tower A to tower C).

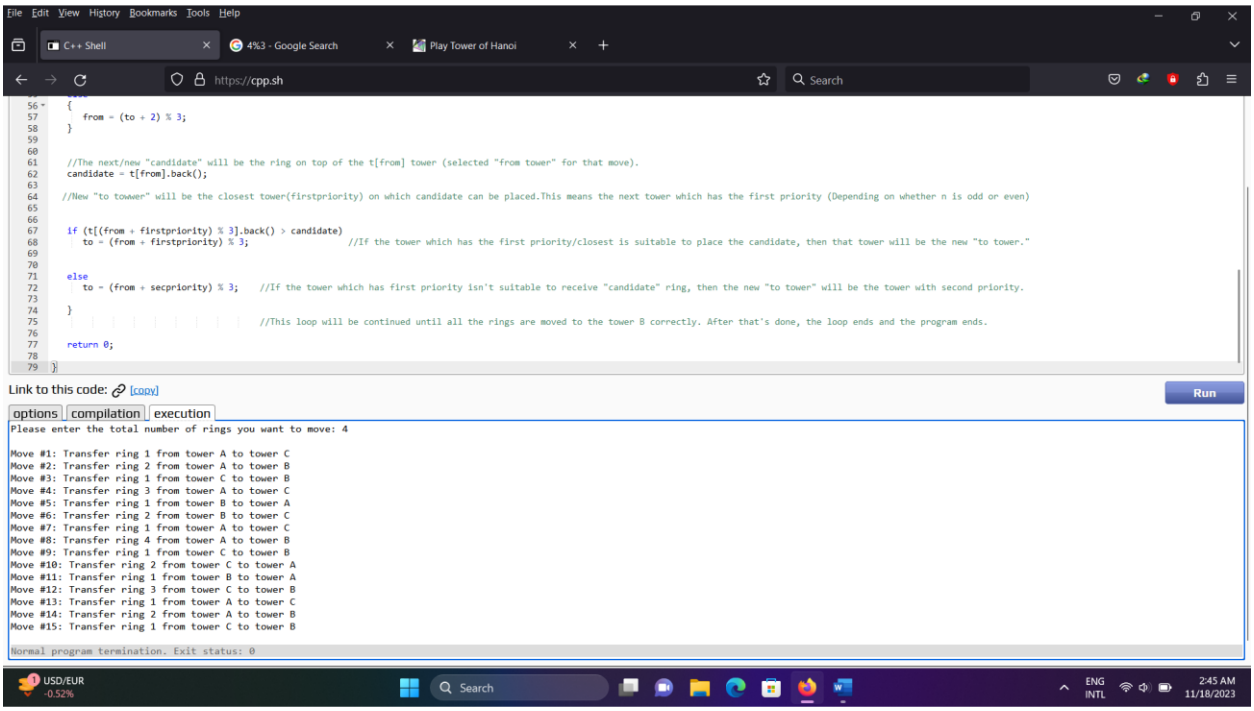
We set the condition of the while loop in the program to "t[1].size() < n+1" so that, when the size of t[1] becomes equal to n+1, which means, when the tower B gets all the rings, this while loop will be terminated, the program will also come to an end as we've found the desired solution.

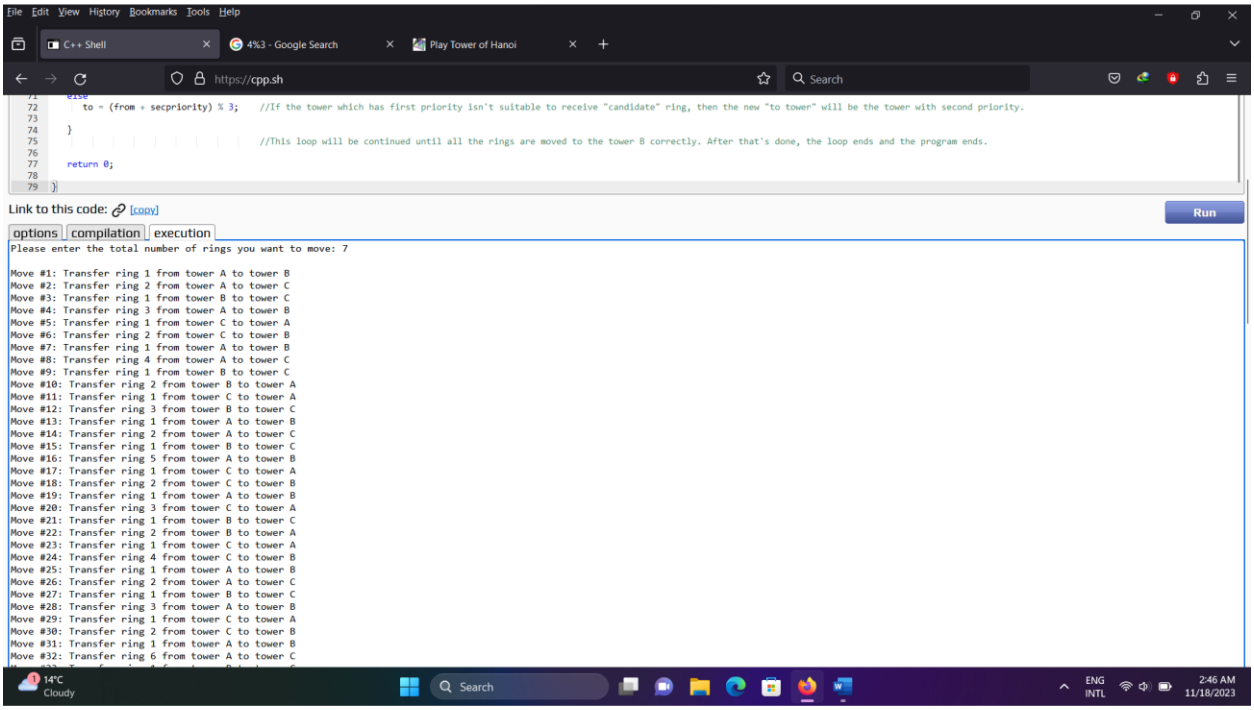
Comment: This is my algorithm & summary to solve the "Towers of Hanoi" problem. I compiled and ran the code; it ran successfully and printed correct solutions of moves. Minimum total move always should be: (2^n) – 1. For example, if we have a total of 4 rings, the minimum total necessary move to solve the puzzle should be (2^4) – 1 = 16 – 1 = 15.

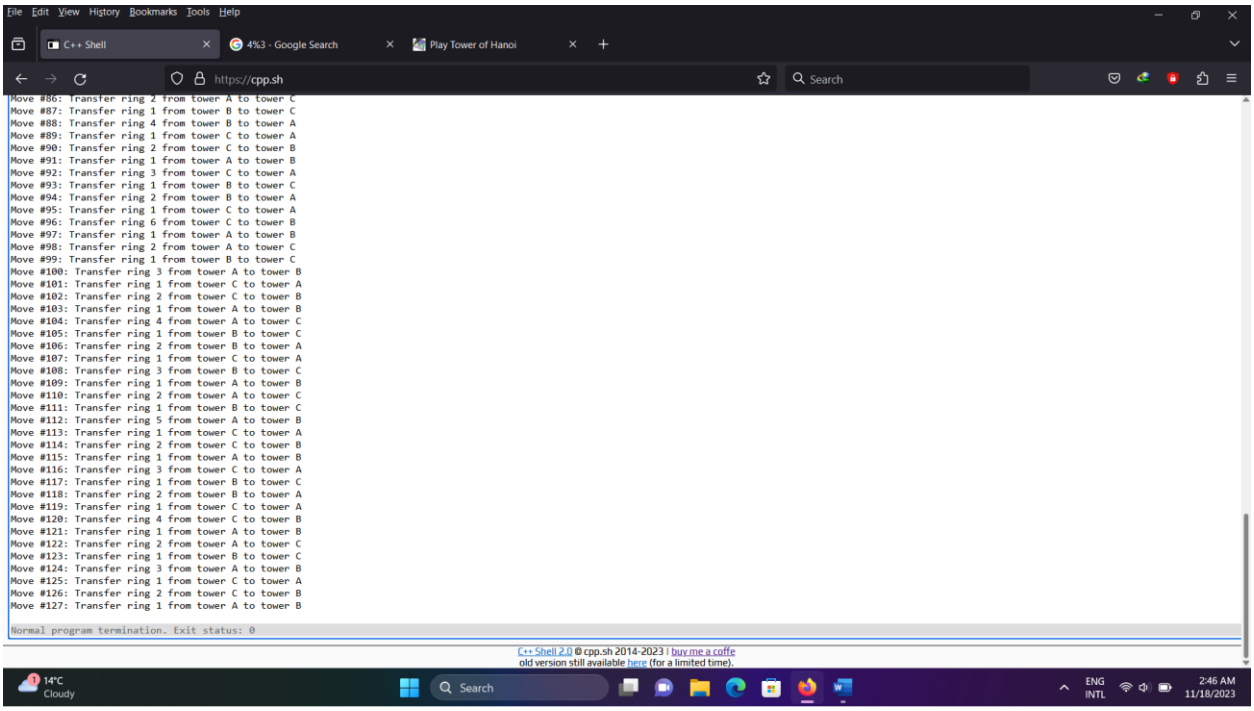
Screenshots of Output:



0


File Edit View History Bookmarks Tools Help
C++ Shell
4%3 - Google Search
Play Tower of Hanoi
https://cpp.sh
Search
{
from = (to + 2) % 3;
}
//The next/new "candidate" will be the ring on top of the t[from] tower (selected "from tower" for that move).
candidate = t[from].back();
//New "to towwer" will be the closest tower(firstpriority) on which candidate can be placed.This means the next tower which has the first priority (Depending on whether n is odd or even)
if (t[(from + firstpriority) % 3].back() > candidate)
to = (from + firstpriority) % 3; //If the tower which has the first priority/closest is suitable to place the candidate, then that tower will be the new "to tower."
else
to = (from + secpriority) % 3; //If the tower which has first priority isn't suitable to receive "candidate" ring, then the new "to tower" will be the tower with second priority.
}
//This loop will be continued until all the rings are moved to the tower B correctly. After that's done, the loop ends and the program ends.
return 0;
}
Link to this code: [copy]
Run
options
compilation
execution
Please enter the total number of rings you want to move: 4
Move #1: Transfer ring 1 from tower A to tower C
Move #2: Transfer ring 2 from tower A to tower B
Move #3: Transfer ring 1 from tower C to tower B
Move #4: Transfer ring 3 from tower A to tower C
Move #5: Transfer ring 1 from tower B to tower A
Move #6: Transfer ring 2 from tower B to tower C
Move #7: Transfer ring 1 from tower A to tower C
Move #8: Transfer ring 4 from tower A to tower B
Move #9: Transfer ring 1 from tower C to tower B
Move #10: Transfer ring 2 from tower C to tower A
Move #11: Transfer ring 1 from tower B to tower A
Move #12: Transfer ring 3 from tower C to tower B
Move #13: Transfer ring 1 from tower A to tower C
Move #14: Transfer ring 2 from tower A to tower B
Move #15: Transfer ring 1 from tower C to tower B
Normal program termination. Exit status: 0
USD/EUR
-0.52%
Search
ENG
INTL
2:45 AM
11/18/2023


File Edit View History Bookmarks Tools Help
C++ Shell
4%3 - Google Search
Play Tower of Hanoi
https://cpp.sh
Search
to = (from + secpriority) % 3; //If the tower which has first priority isn't suitable to receive "candidate" ring, then the new "to tower" will be the tower with second priority.
}
//This loop will be continued until all the rings are moved to the tower B correctly. After that's done, the loop ends and the program ends.
return 0;
}
Link to this code: [copy]
Run
options
compilation
execution
Please enter the total number of rings you want to move: 7
Move #1: Transfer ring 1 from tower A to tower B
Move #2: Transfer ring 2 from tower A to tower C
Move #3: Transfer ring 1 from tower B to tower C
Move #4: Transfer ring 3 from tower A to tower B
Move #5: Transfer ring 1 from tower C to tower A
Move #6: Transfer ring 2 from tower C to tower B
Move #7: Transfer ring 1 from tower A to tower B
Move #8: Transfer ring 4 from tower A to tower C
Move #9: Transfer ring 1 from tower B to tower C
Move #10: Transfer ring 2 from tower B to tower A
Move #11: Transfer ring 1 from tower C to tower A
Move #12: Transfer ring 3 from tower B to tower C
Move #13: Transfer ring 1 from tower A to tower B
Move #14: Transfer ring 2 from tower A to tower C
Move #15: Transfer ring 1 from tower B to tower C
Move #16: Transfer ring 5 from tower A to tower B
Move #17: Transfer ring 1 from tower C to tower A
Move #18: Transfer ring 2 from tower C to tower B
Move #19: Transfer ring 1 from tower A to tower B
Move #20: Transfer ring 3 from tower C to tower A
Move #21: Transfer ring 1 from tower B to tower C
Move #22: Transfer ring 2 from tower B to tower A
Move #23: Transfer ring 1 from tower C to tower A
Move #24: Transfer ring 4 from tower C to tower B
Move #25: Transfer ring 1 from tower A to tower B
Move #26: Transfer ring 2 from tower A to tower C
Move #27: Transfer ring 1 from tower B to tower C
Move #28: Transfer ring 3 from tower A to tower B
Move #29: Transfer ring 1 from tower C to tower A
Move #30: Transfer ring 2 from tower C to tower B
Move #31: Transfer ring 1 from tower A to tower B
Move #32: Transfer ring 6 from tower A to tower C
14°C
Cloudy
Search
ENG
INTL
2:46 AM
11/18/2023


File Edit View History Bookmarks Tools Help
C++ Shell
4%3 - Google Search
Play Tower of Hanoi
https://cpp.sh
Search
Move #86: Transfer ring 2 from tower A to tower C
Move #87: Transfer ring 1 from tower B to tower C
Move #88: Transfer ring 4 from tower B to tower A
Move #89: Transfer ring 1 from tower C to tower A
Move #90: Transfer ring 2 from tower C to tower B
Move #91: Transfer ring 1 from tower A to tower B
Move #92: Transfer ring 3 from tower C to tower A
Move #93: Transfer ring 1 from tower B to tower C
Move #94: Transfer ring 2 from tower B to tower A
Move #95: Transfer ring 1 from tower C to tower A
Move #96: Transfer ring 6 from tower C to tower B
Move #97: Transfer ring 1 from tower A to tower B
Move #98: Transfer ring 2 from tower A to tower C
Move #99: Transfer ring 1 from tower B to tower C
Move #100: Transfer ring 3 from tower A to tower B
Move #101: Transfer ring 1 from tower C to tower A
Move #102: Transfer ring 2 from tower C to tower B
Move #103: Transfer ring 1 from tower A to tower B
Move #104: Transfer ring 4 from tower A to tower C
Move #105: Transfer ring 1 from tower B to tower C
Move #106: Transfer ring 2 from tower B to tower A
Move #107: Transfer ring 1 from tower C to tower A
Move #108: Transfer ring 3 from tower B to tower C
Move #109: Transfer ring 1 from tower A to tower B
Move #110: Transfer ring 2 from tower A to tower C
Move #111: Transfer ring 1 from tower B to tower C
Move #112: Transfer ring 5 from tower A to tower B
Move #113: Transfer ring 1 from tower C to tower A
Move #114: Transfer ring 2 from tower C to tower B
Move #115: Transfer ring 1 from tower A to tower B
Move #116: Transfer ring 3 from tower C to tower A
Move #117: Transfer ring 1 from tower B to tower C
Move #118: Transfer ring 2 from tower B to tower A
Move #119: Transfer ring 1 from tower C to tower A
Move #120: Transfer ring 4 from tower C to tower B
Move #121: Transfer ring 1 from tower A to tower B
Move #122: Transfer ring 2 from tower A to tower C
Move #123: Transfer ring 1 from tower B to tower C
Move #124: Transfer ring 3 from tower A to tower B
Move #125: Transfer ring 1 from tower C to tower A
Move #126: Transfer ring 2 from tower C to tower B
Move #127: Transfer ring 1 from tower A to tower B
Normal program termination. Exit status: 0
C++ Shell 2.0 © cpp.sh 2014-2023 | buy me a coffe
old version still available here (for a limited time).
14°C
Cloudy
Search
ENG
INTL
2:46 AM
11/18/2023