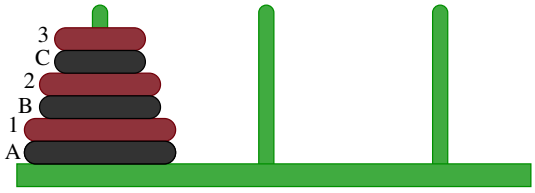
csce350 — Data Structures and Algorithms
Fall 2020 — Project 1: The Tower of Huger

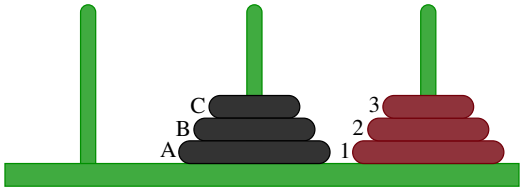
Assigned: September 5
Due: September 21, 11:55pm

The purpose of this assignment is to give you some practice translating a pseudocode description of a recursive algorithm into working software.

The Problem Most computer scientists are familiar with the Tower of Hanoi problem, but not many are familiar with its close cousin, the Tower of Huger problem. In the Tower of Huger, each disk has a *color* — either garnet or black— in addition to its size. The disks come in pairs, with one disk of each color in each size. We measure the input size n as the number of pairs of disks. Here's what the starting state looks like, for $n = 3$:



Each disk has a name, capital letters for black disks and positive integers for the garnet ones, starting with the largest disks. Notice the names in the picture above. Just like in the Tower of Hanoi problem, we can move one disk at a time, but can never stack a larger disk on top of a smaller one. The goal is to separate everything into two separate stacks, one with all the garnet disks and one with all the black disks, like this:



Before continuing to the next page, you might want to think a bit about what sort of algorithm you might use to solve this problem. (Hint: How could you use the SOLVETOWER algorithm for the Tower of Hanoi as a subroutine?)

(This space left intentionally blank to prevent spoilers while you think about the problem.)

Here's a algorithm to solve the Tower of Huger problem.

```
SOLVEHUGER(n, a, b, c)
  if n < 2 then
    Move a disk from a to c.
    Move a disk from a to b.
  else
    MOVESTACKOFPAIRS(n − 1, a, b, c)
    Move a disk from a to c.
    Move another disk from a to c.
    MOVESTACKOFPAIRS(n − 1, b, a, c)
    Move disk from c to b.
    SOLVEHUGER(n − 1, a, b, c)
  end if
```

The function call to MOVESTACKOFPAIRS, which is intended to move a stack of n alternating garnet/black pairs, looks like this:

```
MOVESTACKOFPAIRS(n, a, b, c)
  if n < 2 then
    Move a disk from a to c.
    Move a disk from a to b.
    Move a disk from c to b.
  else
    MOVESTACKOFPAIRS(n − 1, a, b, c)
    Move a disk from a to c.
    Move another disk from a to c.
    MOVESTACKOFPAIRS(n − 1, b, a, c)
    Move a disk from c to b.
    Move another disk from c to b.
    MOVESTACKOFPAIRS(n − 1, a, b, c)
  end if
```

(We could have used the standard SOLVETOWER instead of MOVESTACKOFPAIRS for this purpose. That would have been correct, but slower.)

Your Task Your job for this project is to write a program that solves the Tower of Huger problem, using the SOLVEHUGER algorithm. Specifically, you should write a C++ program that does precisely these things:

1. Read a positive integer n from standard input.
2. Output, to standard output, the starting state of the Tower of Huger with n pairs of disks. The first line should be `Starting at`, and the next three lines should show contents of the three pegs. See example output for details.
3. Output the sequence of moves used by the algorithm above for that value of n to solve the Tower of Huger problem. For each move, describe the move like this,

   `Step __: Move disk __ from peg __ to peg __.`

   showing the move number, the disk that's being moved, the source peg, and the target peg, Then show the state of the pegs, just like at the starting state. Again, be sure to match the exact format of the sample output shown below.
4. Output `Done!`
5. Return to Step 1 above to read and process another value of n. Terminate when standard input reaches end-of-file.

Here's a simple example input and output.

Sample Input 1

```
1
```

Sample Output 1

```
Starting at
 0:A1
 1:
 2:
Step 1: Move disk 1 from peg 0 to peg 2.
 0:A
 1:
 2:1
Step 2: Move disk A from peg 0 to peg 1.
 0:
 1:A
 2:1
Done!
```

Since larger values of n lead to very long outputs, additional sample input/output pairs are available on the course website. You'll want to check them carefully.

Notes A few possibly helpful comments:

- For calibration purposes, my solution has 80 lines of code. If you find yourself writing huge amounts of code, something has probably gone wrong.
- Keep in mind that you are expected to submit your own original work for this problem. Accessing reference material for C++ in general is fine and encouraged. However, you are strongly discouraged from conducting web searches for code specific to the Tower of Huger or Tower of Hanoi problems.

What to Submit You should submit, using the department's dropbox website, a single C++ source file containing all of the code for your program. We will compile this program using this command line:

```
g++ -Wall -std=c++11 yourfile.cpp
```