CSC148, Assignment #1
due Feb 17th, 2017, 10:00 p.m.

From an early draft of his *Canterbury tales*, Chaucer removed an account of the pilgrims staying at Anne Hoy's[1] inn, an establishment that served bad ale, but good cheese. The missing account explained how Anne kept her high-quality cheese stacked on stools, the largest rounds underneath the smaller rounds, to stop rats and mice from getting at them.

Occasionally the stool holding the cheese would need some maintenance (for example, the legs would start to buckle under the weight), and Anne would shift the entire stack from one stool to another. Since she could only move a single (hundred-pound plus) round of cheese at one time, and she refused to stack a larger-diameter cheese round on a smaller one (the outer part of the larger cheese would droop) she used three stools: one was her destination for her entire stack of cheese, one was the source (which likely needed its legs reinforced), and the third was for intermediate stacking. Chaucer immortalized the complicated routine Anne endured, lugging rounds of cheese from stool to stool as "The tour of Anne Hoy " (TOAH).[2]

One of Chaucer's pilgrims had a mathematical bent. She had seen a miraculous early draft of the Wikipedia article on the *Tower of Hanoi* in a vision, and noticed that Anne's routine for moving cheeses was identical to the problem of moving rings between three posts. Using this similarity she calculated that to move n cheeses in this way required $2^n - 1$ moves. This disheartened Anne, who had plans to increase her stack of cheese beyond the 8 she currently had. She decided to invest some of her profits in a fourth stool.

Anne figured that she could do substantially better than $2^n - 1$ moves using the following strategy:

- For a stack of 1 cheese round, her four-stool configuration allowed her to move the stack in 1 move, using her previous three-stool TOAH method.
- To move a stack of $n > 1$ cheese rounds from some origin stool to some destination stool, she reasoned that she could think of some number i between 1 and $n - 1$, and then:
  1. Move $n - i$ cheese rounds to an intermediate stool using all four stools.
  2. Move i cheese rounds from the origin stool to the destination stool, using the (now) only three available stools and her TOAH method.
  3. Move the $n - i$ smallest cheese rounds from the intermediate stool to the destination stool, using all four stools

Notice that steps 1 and 3 require Anne to know how to move $n - i$ cheese rounds using four stools. Anne figured this wasn't a problem, since she could apply her recursive strategy to this smaller problem. She presented her plan to the above-mentioned mathematically-inclined pilgrim who said that if she called the minimum number of moves that her strategy needed to move n rounds of cheese $M(n)$, and if some i between 1 and $n - 1$ were chosen, then (reasoning recursively):

$$M(n) = \begin{cases} 1 & n == 1 \\ 2 * M(n-i) + 2^i - 1 & \text{otherwise.} \end{cases} \qquad (1)$$

After experimenting a bit Anne found she could move 3 cheese rounds in 5 moves (a little better than the 7 required by the TOAH method), and 6 cheese rounds in 17 moves — much better than the 63 required

[1] In Middle English her name would have been spelled *Auyne H'Oeuil*

[2] Chaucer conjectured Anne's muttered "cheeses crust!" and "gentle jumping cheeses!" were veiled religious oaths, not mundane references to cheese maintenance.

by the TOAH method. But the choice of i made all the difference. She (and the aforementioned math-geek pilgrim, who had decided to stay at her inn permanently) spent many hours with early prototypes of pencil and paper, figuring out the best strategies for moving ever-larger stacks of cheese.

This is where matters stood, for centuries, until the invention of the computer.

your job

The *Tour of Anne Hoy Game* is to move a stack of cheeses from the first stool to the last stool, using only valid moves as described in the previous paragraphs.

You will implement parts of a step-by-step design of an object-oriented program for a *Tour of Anne Hoy game*. In order to guide you, we present the steps below, in the order we recommend. Notice that you get credit for each step you complete.

We know that this is a challenging assignment. We aim to help guide you to a successful submission, provided you follow the guidance we offer. You may work on this assignment in groups of 1, 2, or at most 3. You should be familiar with and understand all the code that your group submits . . . or you'll pay for it come test time.

We encourage you to submit your work to MarkUs frequently, especially each time you complete one of the steps below.

Step 1: Read and understand the `Cheese` class in toah_model.py, and implement the methods where we have given headers (don't change the headers). This step is worth 5%.

Step 2: Read all client code in gui_controller.py, gui_viewables.py, console_controller.py, and tour.py, to see how it uses class `TOAHModel`. Unlike `ConsoleController`, `GUIController` is already completely implemented for you. You do not need to understand (and you certainly should **not** change) the code in the files gui_controller.py and gui_viewables.py, but your implementation of `TOAHModel` will be expected to work with both `ConsoleController` (once you complete step 4) and `GuiController`.

Write the method headers for `TOAHModel` in toah_model.py according to their uses in client code. You may need to add some methods to be used for implementations you write for `ConsoleController` and in module `tour`. Note that these method headers, like all method headers, must contain docstrings. They include type contract (written in the CSC108 way from last semester or the annotation method that we introduced this semester), description, and examples.

Step 2 is worth 45% of the credit for this assignment (of course, you'll need to have done Step 1).

Step 3: Write the implementations of the `TOAHModel` methods from Step 2. Notice in particular how `ConsoleController` and `GUIController` rely on a `TOAHModel` object to throw exceptions in certain circumstances (and without altering the state of the `TOAHModel` object).

When you're done, you should be able to run gui_controller.py to manually play the *Tour of Anne Hoy* game. Do not change anything in gui_controller.py or gui_viewables.py.

You also need to implement `TOAHModel.__eq__`. The header in toah_model.py has instructions.

You can assume that no two `Cheese` objects of the same size will ever be added to a `TOAHModel` object.

Steps 1–3 are worth 65% of the credit for this assignment.

Step 4: Now we want you to get a console-based version of the game working. Implement the methods `play_loop, __init__`, and the function `move` in console_controller.py. The headers are already written for you, and the docstrings provide additional instructions. You will also need to put appropriate code under `if __name__ == "__main__":`.

When you're done, you should be able to run console_controller.py to manually play the *Tour of Anne Hoy* game through the console.

Steps 1–4 are worth 70% of the credit for this assignment.

Step 5: Implement the module-level function `tour_of_four_stools` in tour.py to solve four stool instances of Anne Hoy's cheese-moving problem, trying to minimize the number of moves. You should be able to move n cheeses in considerably less than $2^n - 1$ moves achievable with just three stools.

We will check your solution using `TOAHModel.get_move_seq`, so make sure to use the class `MoveSequence` in toah_model.py to record the moves that your solver makes. If we cannot verify your sequence of moves using `TOAHModel.get_move_seq`, then we cannot give you marks for this step.

Step 5 is worth 22% of the credit for this assignment. Any solution of the game (that doesn't take, say, more than twice as long as the 3-stool towers-of-Hanoi strategy) will get 3%. If you solve it in fewer moves than required by the 3-stool strategy (fewer than $2^n - 1$), we'll give you an additional 7%. If your solution is close to as good as the best-known solution, which we described above, then you'll get an additional 8%. If you implement the best-known solution, you'll get an additional 4%.

Step 6: Add an option to your `tour` module that enables animating your solver in the console. The header for `tour.tour_of_four_stools` already has two optional arguments for this; when `console_animate` is `True`, animation should be displayed in the console, and `delay_btw_moves` gives the number of seconds to wait between showing two moves. We recommend using `TOAHModel.__str__`, but this isn't required (you can make your own string-based representation).

Step 6 is worth 8% of the credit for this assignment.
Steps 1-6 are worth 100% of the credit for this assignment.

Those licenses: Notice that the starter code that we provide is distributed with a GNU GPL license. One consequence of this is that any copies of that code, or any work you derive from it (for example the files you submit to MarkUs), must be similarly licensed if you pass it on to someone else — you may just add your name to the license declaration, and include the COPYING notice when passing it along.

what to submit

You'll need to submit your version of the following files to MarkUs:

- `toah_model.py`
- `tour.py`
- `console_controller.py`
- We will deduct up to 10% for files that do not pass the `python_ta.check_all(...)` code that we include at the end of your starter code The `python_ta.check_all(...)` directives include `config=...` to specify a configuration file saying which `python_ta` warnings to disable. You may **not** change the config files.