CS 116 Winter 2015

Assignment 05

Due at Noon on Wednesday, February 25

Assignment Guidelines:

- **This assignment consists mostly of material as covered in Module 05. All solutions must be in Python.**
- Your solutions should be placed in files **a05qY.py**, where Y is a value from 1 to 4.
- **Download the testing module from the course web page. Include `import check` in each solution file.**
    - **When a function produces a floating point value, you must use `check.within` for your testing. Unless told otherwise, you may use a tolerance of 0.0001 in your tests.**
- Note the following new restrictions:
    - **Do not import any modules other than `math` and `check`.**
    - **Do not define helper functions locally. Define all helper functions above the required function.**
    - **Examples and tests are not required for any helper functions.**
    - **Do not use Python iteration (loops) or lists. All repetition should be implemented using recursion.**
    - **Do not use any other Python functions not discussed in class or explicitly allowed elsewhere.** See the allowable functions post (#19) on Piazza. You are always allowed to define your own helper functions, as long as they meet the assignment restrictions.
    - **You may use global constants in your solutions.**
    - **Do not use global variables for anything other than testing.**
- Download the interface file from the course Web page to ensure that all function name are spelled correctly and each function has the correct number and order of parameters. Use the function headers in your submitted files for each question.
- For full style marks, your program must follow the **Python section** of the CS116 Style Guide. In particular,
    - Be sure to include all the steps of the design recipe for all required functions: including purpose statements, contracts (including requirements), **examples (note the new style)**, and tests. **No functions on this assignment should require an effects statement.**
    - You are not required to submit any templates with your solutions.
    - The purpose should be written in your own words and must include meaningful use of the function's parameter names.
    - There will be marks assigned for the appropriate use of constants, helper functions, choice of meaningful names, and appropriate use of whitespace.
- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.
- Do not send any code files by email to your instructors or ISAs. It will not be accepted by course staff as an assignment submission, even if you are having issues with MarkUs. Course staff will not debug code emailed to them.
- Test data for all questions will always meet the stated assumptions for consumed values.
- Read each question carefully for specific restrictions.
- No late assignments will be accepted.
- Check MarkUs and your basic test results to ensure that your files were properly submitted. In particular:
    - A misnamed file or function will receive no marks.
    - Do not copy any text from the interactions window in WingIDE or from this pdf into your programs (even as comments). It will make your submitted file unreadable in MarkUs and you will receive no marks (correctness or style) for that question.
- Be sure to review the Academic Integrity policy on the Assignments page.

Coverage: Module 5
Language level: Python 3

CS 116 Winter 2015

Assignment 05

Due at Noon on Wednesday, February 25

1. The integers *{a_1,a_2,a_3, …}* form an arithmetic sequence if the difference between any two consecutive values (i.e. $a_{j+1} - a_j$, for *j=1,2,3,…*) is always the same. For example, *{2,4,6}* are the first 3 integers in an arithmetic sequence since the difference between 2 and 4 is the same as the difference between 4 and 6., However, *{-4,-1,5}* is not since the difference between -4 and -1 is not the same as the difference between -1 and 5.

   Write a Python function `next_in_sequence` that consumes three integers (called `a`,`b`,`c`). If `a`,`b`,`c` form the start of an arithmetic sequence, the function produces the next number in the sequence. If they do not, the function produces `False`. For example,
   - `next_in_sequence(2,4,6) => 8`
   - `next_in_sequence(40,0,-40) => -80`
   - `next_in_sequence(-4,-1,5) => False`

2. Write a Python function `participation` that consumes 5 natural numbers (called `correct`, `incorrect`, `unanswered`, `threshold`, and `tutorials_attended`) and produces a floating point number between 0 and 5, corresponding to the participation grade for a CS116 student who had `correct` clicker questions answered correctly, `incorrect` questions answered incorrectly, `unanswered` questions not answered (with the best `threshold` % of questions counted), and who attended `tutorials_attended` tutorials. When calculating with `threshold`, use the `math.ceil` function to round up the number of questions to count. Each correct answer is worth 2 points, each incorrect is worth 1 point, and each unanswered is worth 0 points. Each tutorial attended is worth a bonus of 0.1 marks on top of the clicker question grade (out of 5). The total participation grade cannot exceed 5.0.

   For example,
   - `participation(4,3,0,100,0) => 3.92857` (approximately)
   - `participation(2,3,2,80,7) => 3.61667` (approximately)
   - `participation(3,2,2,50,12) => 5.0`

   Notes:
   - Assume that
     - `correct + incorrect + unanswered > 0`,
     - `0 < threshold <= 100`,
     - `0 <= tutorials_attended <= 12`.
   - While this question is similar to Question 2 from Assignment 01, be sure to note the differences in parameters and in the produced value.
   - The number of places after the decimal displayed in your IDE may not match those shown above. That doesn't matter. What matters is whether your function passes your tests. You do not need to round the results to any set number of digits.

3. As all CS116 students know by now, the Fibonacci sequence is a well-known mathematical sequence (with lots of interesting, real-life applications) and is defined as:

$$F_n = \begin{cases} 0, & if\ n = 0 \\ 1, & if\ n = 1 \\ F_{n-1} + F_{n-2}, & if\ n > 1 \end{cases}$$

   In Module 03, we studied three implementations to calculate the n^{th} Fibonacci number using Scheme. In this question, you will investigate two implementations to calculate the n^{th} Fibonacci number using Python.

Assignment 05
Due at Noon on Wednesday, February 25

a) Write a Python function `fib_rec` that consumes a `Nat`, called `n`, and uses **accumulative recursion** to calculate and return the nth Fibonacci number.

As discussed in class, there are limitations to the depth of recursion that can be called in Python, so you must use the accumulative approach illustrated in the `fib3` function in Module 03 or your implementation will only work for relatively small values of `n`. In addition, Fibonacci numbers grow very quickly. Therefore, we will not test your `fib_rec` function for any `n` > 100.

b) There is also a closed form formula for calculating the nth Fibonacci number, which avoids recursion entirely:

$$F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}, where\ \varphi = \frac{1+\sqrt{5}}{2}.$$

Write a Python function `fib_approx` that consumes a `Nat`, called `n`, and uses the above formulas to produce an approximation to the nth Fibonacci number. Leave the approximation of φ (pronounced "phi") as a floating point value, but use the `int` conversion operation on the fractional form of F_n in the formula.

Now, in theory and with exact arithmetic, the fraction above would equal F_n. In practice, it will produce the actual Fibonacci number only up to some point, at which point, due to limited precision, some of the leading digits will still be correct, but the trailing digits will be wrong. By trial and error (and by using your solution to part (a) or available lists of Fibonacci numbers) determine the value `M`, such that `fib_approx(n)` produces the correct value for `n <= M`, and produces the incorrect answer starting with `n = M+1`. Include this value `M` with your contract requirements for `fib_approx`. *You do not need to prove this result, just base it on your observations.*

4. There are some cool mathematical tricks for determining if a natural number *n* is a multiple of various values less than 10. For example, consider the following steps to determine if *n* is a multiple of 7:
    a. If *n* is 0 or 7, it is a multiple of 7
    b. If n ≤ 10, but not 0 or 7, it is not a multiple of 7
    c. Otherwise:
        i. Determine the last digit of *n* and double it
        ii. Subtract the *doubled last digit* from the *rest* of the digits of *n* and take the absolute value (call this new value $\hat{n}$).
        iii. *n* is a multiple of 7 if $\hat{n}$ is a multiple of 7 (which is determined by repeating all the steps from (a) with $\hat{n}$).

For example,

- Consider n = 1234. First calculate 123 – 2*4 to get 115. Then 11-2*5 gives 1. Since 1 is not a multiple of 7, we know 1234 is not a multiple of 7.
- Consider n = 35. First calculate 3 – 2*5 to get -7, whose absolute value is 7. Since 7 is a multiple of 7, we know 35 is a multiple of 7.

Write a Python function `mult_of_7` that consumes a `Nat`, called `n`, and, using the steps of the recursive algorithm described above, produces `True` if `n` is a multiple of 7 and `False` if it is not. For example, `mult_of_7(1234) => False`.

Assignment 05

Due at Noon on Wednesday, February 25

Note: You must use the steps described above to solve this problem in order to receive any correctness marks.