Assignment Guidelines:

- This assignment covers material up to Module 2.

Submission details:
- Solutions to these questions must be placed in files a02q1.py, a02q2.py, a02q3.py, and a02q4.py, respectively, and must be completed using Python 3.
- Download the interface file from the course website to ensure that all function names are spelled correctly and each function has the correct number and order of parameters.
- All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
- Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
- For full style marks, your program must follow the Python section of the CS116 Style Guide.
- Be sure to review the Academic Integrity policy on the Assignments page.
- Natural numbers in this course begin at 0.
- Required functions need all design recipe elements. Functions you define (e.g. helper functions) need all design recipe elements except for examples and tests.

Download the testing module from the course website. 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 0.00001 in your tests.
- Test data for all questions will always meet the stated assumptions for consumed values.

Restrictions:
- Do not import any modules other than math and check.
- You are always allowed to define your own helper/wrapper functions, as long as they meet the assignment restrictions. Do not use Python constructs from later modules (e.g. dictionaries, loops (for or while or others), zip, sorted, anything with sets or enumerators, slicing, indexing (square brackets), string methods and/or lists). Use only the functions and methods as follows:
  * abs, len, max and min
  * Any method or constant in the math module
  * Any basic arithmetic operation (including +, -, *, /, //, %, **) 
  * Any basic logical operators (not, and, or)
  * Typecasting including int(), str(), float(), bool()
  * if statements
  * Recursion
- While you may use global constants in your solutions, do not use global variables for anything other than testing.
- Read each question carefully for additional restrictions.
- 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.
1 Grade Computation

In this problem, we will write a function to help compute your CS 116 grade. The basic grade calculation for CS116 is:

\[ 20\% \text{ Assignments} + 30\% \text{ Midterm} + 45\% \text{ Final} + 5\% \text{ Participation} \]

where Assignments, Midterm, Final and Participation are grades between 0 and 100 inclusive, corresponding to grades for those course components. However, this calculation is used only if a student passes the weighted exam average, that is, if and only if

\[
\frac{30\% \cdot \text{Midterm} + 45\% \cdot \text{Final}}{75} \cdot 100 \geq 50
\]

Write a function

\[ \text{cs116\_grade(assign, mid, fin, part)} \]

that consumes assign, mid, fin, part (four natural numbers between 0 and 100, inclusive, corresponding to a student’s grade in assignments, midterm exam, final exam, and participation, respectively). If a student passes the weighted exam average, the function produces the result of the basic calculation described above. If the student does not pass the weighted exam average, the function produces the smaller of the basic calculation and the weighted exam average. Note: the calculations described above will produce a floating point value for the grade. Use the built-in function \text{round} to convert your final answer to the nearest integer value. Do not perform any rounding on intermediate results.

Sample:

\[
\begin{align*}
\text{cs116\_grade}(75,60,80,90) & \Rightarrow 74 \\
\text{cs116\_grade}(80,52,40,100) & \Rightarrow 45 \\
\text{cs116\_grade}(0,55,60,0) & \Rightarrow 44 \\
\end{align*}
\]

2 Goods Tax

In Canada, most goods have a tax that is added to a price of a good at the checkout. There are two main types of taxes, a provincial tax (PST) and a federal tax (GST). The GST tax in Canada is set at 5%. The amount of tax depends on the region where you live. Provincial taxes are listed below

- In Alberta, Northwest Territories, Nunavut, and Yukon (abbreviations AB, NT, NU, YT), there is no additional provincial tax.
- In Saskatchewan (abbreviation SK), the provincial tax is 6%
- In British Columbia (abbreviation BC), the provincial tax is 7%
- In Manitoba and Ontario (abbreviations MB, ON), the provincial tax is 8%
- In Quebec (abbreviation QU), the provincial tax is 9.975%
- In New Brunswick, Newfoundland and Labrador, Nova Scotia and Prince Edward Island (abbreviations NB, NL, NS, PE), the tax is 10%

Some goods are exempt from certain taxes. The situation is vastly more complicated than what we will go through here but we simplify for convenience:

- Food is exempt from all taxes
- Literature and child related goods are tax exempt from PST but not from GST.
- Everything else has both taxes applied.

Your goal is to write a function:

\[ \text{after\_tax(cost, prov\_abbr, item)} \]

\footnote{There is also an HST but this is a merger of the two taxes and isn’t relevant for our discussions}
which returns the selling price of the item, a string, based on the cost and with the appropriate tax applied based on where you live in prov_abbr and whether or not the item should be taxed. You may assume that tax exemptions only apply if item is one of "food", "literature" or "child". The province abbreviations are given above as well and you may assume that prov_abbr is one of those abbreviations.

Sample:

after_tax(20.0, "ON", "literature") => 21.0
after_tax(20010.0, "AB", "car") => 21010.5

3 Jeopardy!

In the classic game show Jeopardy!, three contestants compete to answer trivia questions and earn money by doing so. The person at the end of the game with the most money gets to keep what they have earned.

At the last portion of the show, contestants must make a secret bet of their winnings thus far before seeing the final question. If they get the question right, they add their bet to their total. If they get it wrong, they lose that amount of money from their total. A common strategy for the person in the lead is to bet exactly enough money such that if they get the question right and the second place person were to bet everything and also get the question correct, the first player would be victorious. An example should help. Suppose our three contestants had 5000, 7500 and 10000 dollars respectively. If the second place person, who has 7500, were to bet all of their 7500 and get the final question correct, they would have a total of 15000. Thus, the first place player would want to bet 5001 so that if both the first place player and the second place player were to get the final question correct, the first place player would win.

Given the three contestants scores, write a function

    final_jeopardy(c1, c2, c3)

which consumes the three natural number scores and returns the amount that the leader should bet in order to guarantee victory if they get the question correct. If two people are tied for first, they should bet everything. If a person is too far ahead to be caught, assume they will bet 0.

Sample:

final_jeopardy(5000, 7500, 10000) => 5001
final_jeopardy(15000, 15000, 800) => 15000

4 Prime Divisors

Write a function

    count_prime_divisors(n)

which returns the number of prime divisors of a given positive integer n. Do not count multiplicity of these divisors. Due to recursion limitations, your code will not be tested on numbers larger than 100. Do not hard code all 100 cases (marks will be deducted for codes that do not use recursion).

Sample:

count_prime_divisors(12) => 2