Assignment 02
Due September 25th at 10:00am (no late submissions)

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.
  - Helper functions need design recipe elements but not 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 functions, as long as they meet the assignment restrictions. Do **not** use Python constructs from later modules (e.g. loops, slicing, indexing, 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 `+, -, *, /, //, %, **`)
    - Type casting 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 Leap Years

Recall that a leap year is a year in which February has an extra day. A leap year is any year that is divisible by 400 or any year that is divisible by 4 and not also divisible by 100.

Write a function:

`is_leap_year(n)`

which consumes a year `n` and returns `True` if and only if the year `n` is a leap year. (For those of you versed in English history, ignore the British Act of 1751 and assume the above definition applies for all years).

Sample:

```
is_leap_year(2004) => True
```

2 ELO Rating

In chess, a common rating system used is the ELO rating system. The exact implementation details can differ slightly from what is presented here but below is the primary idea. In chess, each player has a positive integer ELO rating. [1] Assume two players A and B are playing and each has a rating of R_A and R_B. Based on this, each player has a probability of winning (an expected score) as computed by a logistic curve:

$$E_A = \frac{1}{1+10^{(R_B-R_A)/400}} \qquad E_B = \frac{1}{1+10^{(R_A-R_B)/400}}$$

One should note that if $R_A > R_B$, then $E_A > E_B$ (that is, if R_A is a better player, they should be expected to do better) and that $E_A + E_B = 1.0$. Further, we define a development constant K as follows:

- $K = 10$ if a player's published rating has reached 2400 and remains as $K = 10$ subsequently, even if the rating drops below 2400. This information will be stored in `was_2400` below.
- $K = 40$ for a player new to the rating list for their first 30 games and who has never had a rating of 2400 or more.
- $K = 40$ for all players until their 18th birthday, as long as their rating remains strictly under 2300. A player who reaches 2300 before the age of 18 will fall into the last category below.
- $K = 20$ as long as a player's rating has never exceeded 2400 and they have played at least 30 games and is at least 18 years old.

and note that each player has a score S_A and S_B respectively which corresponds to one of 1.0, 0.5 or 0.0 for a win, draw or loss respectively (so that again $S_A + S_B = 1.0$). With these value, the new ratings are computed as follows:

$$R_{A_{new}} = R_A + K(S_A - E_A) \qquad R_{B_{new}} = R_B + K(S_B - E_B)$$

Given R_A, R_B, S_A, the age of player A in `age_of_a`, `was_2400` (if player A was ever at least 2400), and the number of games played stored in `num_games`, write a function

`new_rating(ra, rb, sa, age_of_a, was_2400, num_games)`

that computes player A's new rating as described above. You should take the floor of the final rating to convert any fractional rating to an integer. If a calculation results in a negative rating, the rating of the player should be 0. [2] You may also assume that if `ra >= 2400` then `was_2400` is `True`.

Sample:

[1] If you have never played chess before, there is a way to compute your rating based on your opponent's ratings. There is also a default rating that can be assigned as well.

[2] The actual minimum is 100. Ratings cannot be lower than 100 according to the official standard.

```
new_rating(2350, 2375, 1.0, 19, False, 100) => 2360
```

3 ISBN Check Sum

Books that are issued a 13-digit International Standard Book Number (ISBN) have a special property called a checksum that helps to catch typos made while copying the number. Books are given a 12 digit code and then a thirteenth digit is appended to the end of the number to create a full ISBN. This is done as follows. For the twelve digit number n, take the sum of the digits in the odd positions (starting with the 1st position) and add them to three times the sum of the digits in the even position (starting with the second position). Then take the remainder of this number when divided by 10 to give the last digit. For example, if your number was $n = 567856785678$, adding the odd position digits gives $5+7+5+7+5+7 = 36$ and adding the even positioned digits gives $6+8+6+8+6+8 = 42$. Then, the remainder when I divide $36 + 42 \cdot 3 = 162$ by 10 is 2 which will become the last digit. Thus, your ISBN would be 5678567856782.

Your job is to write a function:

`full_isbn(n)`

which consumes a partial ISBN `n` and returns the thirteen digit ISBN.

Sample:

```
full_isbn(567856785678) => 5678567856782
```

Note that if the number contains leading zeroes, they will not be present (so 012345678912 will be passed as 12345678912). Your final answer should also not have any leading zeroes present if applicable. If coded correctly, your function should be able to handle leading zeroes without any issues.

4 Arithmetic

Write a function

`strange_sum(n)`

that consumes a natural number `n` and computes the following sum:

$$1 + \frac{1}{1} + \frac{1+2}{1\cdot 2} + \frac{1+2+3}{1\cdot 2\cdot 3} + ... + \frac{1+2+...+n}{1\cdot 2\cdot ...\cdot n}$$

The sum above is described as follows. There are $n+1$ total terms and the ith term for each i between 0 and n inclusive has as its numerator the sum of the first i natural numbers and its denominator is the product of the first i natural numbers. Start indexing terms at 0 (where we think of the empty sum and the empty product as 1 respectively so that the zeroth and first terms in the sum are both 1).

Sample:

```
strange_sum(2) => 3.5
```