Assignment Guidelines:

Do NOT use recursion, map, or filter. All repetition must be performed using iteration (while and for loops only). Solutions using recursion or abstract list functions will receive no correctness marks.

- This assignment covers material in Module 6.
- Submission details:
  - Solutions to these questions must be placed in files a06q1a.py, a06q1b.py, a06q2.py, a06q3.py, and a06q4.py, respectively, and must be completed using Python 3.
  - Download the interface file from the course Web page 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 web page. Include import check in each solution file.
  - When a function returns 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.
  - Do not use Python constructs from later modules. See the allowable functions post on Piazza. You are always allowed to define your own helper functions, as long as they meet the assignment restrictions.
  - 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 Revisiting Past Problems

For this question, you will revisit problems tackled on previous assignments. All of the marks will be based on correctness and meeting the stated restrictions. You are encouraged to include other parts of the design recipe but this is not required and no marks will be assigned to style, purposes, effects, contracts, examples and choice of tests.

a) Generalize the last question on Assignment 2 using loops. Write a function `lucky_sevens` which consumes two integers `a` and `b` and returns the number of times the digit 7 would appear if you wrote all the integers between the integers `a` and `b` inclusive. Do not make any assumptions about the size of these integers or this range. For example:

```
lucky_sevens (1, 10) => 1
lucky_sevens (5, 70) => 8
lucky_sevens (76, 78) => 4
lucky_sevens (-77777, 77777) => 77780
```

You may not use strings to solve this problem.

b) Use loops to implement the function `redact_text` from Assignment 4. It consumes two lists of words, `original` and `banned`, and returns the number of times words in `banned` occur in `original`. In addition, the list `original` is mutated so that each occurrence of a banned word in `original` is replaced with the string "<redact>". Note that words in `banned` may occur multiple times in `original`, and the total number of times is the returned value. The list `banned` is not changed by `redact_text`. For example, suppose

```
speech = ["cs", "is", "fun", "homework", "is", "sometimes", "fun", "exams", "are", "not", "fun"].
```

Then `redact_text(speech, ["fun", "is", "not"])` => 6 and updates `speech` to

```
["cs", "<redact>", "<redact>", "homework", "<redact>", "sometimes", "<redact>", "exams", "are", "<redact>", "<redact>"].
```

Recall that for the purposes of this question, we will consider a word to be a nonempty string consisting only of lowercase alphabetic characters.

2 Run Length Encoding

Recall run length encoding from Assignment 3 which is a simple way to compress strings that contain the same character repeated many times in a row. It uses two key ideas:

- a string can be partitioned into blocks (e.g. 'aaabbabbbbbbb' can be partitioned into the blocks 'aaa', 'bb', 'a' and 'bbbbbb'), and
- each block can be represented by a number and one character (e.g. 'aaaaa' can be represented by '5a').

For example, the encoding of 'aaabbabbbbbbb' is '3a2b1a11b'.

More formally, a block is a substring of identical characters that is as long as possible. A block is represented by the length of the block followed by the character. The encoding of a string is the representation of each block in the string in the order in which they appear in the string. The encoding of the empty string is the empty string.

Write a function `encode` that consumes a string where each character is 'a' or 'b', and returns the encoding of the string. For example, `encode('aaabbabbbbbbb')` should return '3a2b1a11b'.
3 Standings

Elmira recently hosted the Ontario curling championships. This problem requires you to take the results of all the games in a tournament like this and determine the team with the most wins.

Specifically, write a function `most_wins` that consumes `T` and `L`. The parameter `T` is a list of distinct strings where `T[i]` is the name of team `i`. The parameter `L` is a nested list containing the results of all the games in the tournament. Specifically, each `L[i][j]` is a two element list `[x, y]` where `x` and `y` are natural numbers. If `x > y`, then we know team `i` beat team `i + j + 1`. Otherwise, we know team `i + j + 1` beat team `i`.

An example is below. To help you visualize this, note that `L` corresponds to the upper right portion of the table shown.

```
HOMAN   AULD   HORTON  TIPPIN
HOMAN   4 - 0  7 - 2  5 - 1
AULD     3 - 6  8 - 1
HORTON
TIPPIN
```

```python
scotties_teams = ['HOMAN', 'AULD', 'HORTON', 'TIPPIN']
scotties_results = [
    [[4,0],[7,2],[5,1]],
    [[3,6],[8,1]],
    [6,4]
]
```

The first row gives the scores in all three games played by 'HOMAN'. The second row gives the scores of the other two games played by 'AULD', and the third row gives the score in the game between 'HORTON' and 'TIPPIN'. These results are:

```
HOMAN 4  AULD 0
HOMAN 7  HORTON 2
HOMAN 5  TIPPIN 1
AULD 3  HORTON 6
AULD 8  TIPPIN 1
HORTON 6  TIPPIN 4
```

Your function must return the name of the team with the most wins. So, continuing this example,

```
most_wins(scotties_teams, scotties_results) => 'HOMAN'
```

A small second example is `most_wins(['EPPING', 'MCDONALD'],[[4,5]]) => 'MCDONALD'

Note:
- There will be at least two teams.
- Every two teams play each other exactly once.
- No game results in a tie.
- There will always be a unique team with the most wins.
4 Histogram

Write a function histogram that prints a histogram of integer grades between 0 and 100 (inclusive) read from the keyboard. It does not consume or produce anything.

The user will enter grades one per line. The end of the data will be indicated by an empty line. That is, after entering each grade, the user will just hit enter causing input to return ''.

Do not print any prompts. Only print a histogram exactly matching the format of the two examples below.

Note:
- Use exactly the intervals shown. An interval \([a, b)\) includes \(a\) but not \(b\).
- Align the closing brackets and follow each with a single space.
- Put a single asterisk beside the correct interval for each grade entered by the user.
- Ensure the last character in each line is either an asterisk or a single space after the closing bracket.