REMINDER

• Midterm is the Monday, March 4th, at 7:00pm – 8:50pm
  – It’s on Monday. STUDY!!

• Look up your seat on Odyssey.

• Additional practice problems are available for each module. See “Additional Materials” on the course web page.
To what extent are you prepared for the midterm?

A. I’m going to rock this midterm.
B. I’m stressed and feeling unprepared for the midterm.
C. Somewhere between A and B
D. WHAT!! There’s a midterm? Since when?!
E. I don’t know if I’m prepared for the midterm or not. 😞
CQ 2

How confident are you on your design recipe?

A. Very confident
B. Not confident at all
C. Somewhere in between A and B
D. A little bit, but I’m still worried.
E. Not at all, and I have given up on it.
CQ 3

Have you been following along on Piazza?

A. Yes! I go through Piazza everyday.
B. Yeah, I go through it regularly.
C. I will check Piazza if I have a question about the course.
D. I don’t check Piazza.
E. What is it?
**DESIGN RECIPE**

- **Contracts:**
  - Use the form
    
    ```
    fun_name: types consumed -> type returned
    ```
  - Use the single arrow in contracts!
  - Make sure you use the correct type names (i.e `Str` not `String`; it’s `Float` not `Num` in Python, etc.)
    
    - Do not pluralize your type names
    - Capitalize your type names

- **Requirements:**
  - Include requirements to any of the types consumed, if it has any.
• Purpose:
  – Make sure you mention all of the parameter names in your purpose and how they relate to what is being returned
  – Keep it short and simple; do not copy directly from the question!
  – Make it clear if you are “returning”, “printing”, “mutating”, or “reading input”
  – Use return, not produce.

• Effects:
  • Be clear and concise on the different effects: printing, input, and mutation
    o Print to the screen
    o Read input
    o Mutate list
• **Examples/Tests:**
  - For examples, make sure to have a base case and a non-base case at minimum
  - **Example Format**
    - **Examples:** `fn_call(x1,x2,...,xn) => expected`
    - **Use double arrows in examples!**
  - **Tests:** `check.expect` and `check.within`
    - `check.set_input` *(when input() is used)*
    - `check.set_screen` *(when information is printed)*
    - An extra `check.expect` to check mutation if the function mutates a list (parameter)
      - `check.expect("label_1", fcn_call(p1,..,pn), expected returned value)`
      - `check.expect("label_1(mutation)", L, expected list after mutation)`
Out of the following topics, which one would be most helpful to cover now in preparation of the midterm?

A. **Conditions**
B. **Strings and/or Input and Output**
C. **Lists**
D. **Recursion**
E. **Anything is fine**
Let's say we have large bricks that are 5 inches in length and small bricks that are 1 inch in length.

Write a function called `enough_bricks` which has three parameters: `small`, the number of small bricks, `large`, the number of large bricks and `goal`, the length of a row we want to build. `enough_bricks` returns True if you can create a row with same length as `goal` with the number of small and large bricks available, False otherwise.

Examples:

- `enough_bricks(3,1,8) => True`
- `enough_bricks(3,1,9) => False`
- `enough_bricks(1,2,9) => False`

Source: Coding Bat, http://codingbat.com/prob/p118406
Write a function called `ends_with_other` that consumes two strings, `s` and `t`, and returns `True` if `s` ends with `t` or if `t` ends with `s`, `False` otherwise. This function should be case insensitive.

Examples:
- `ends_with_other("abc", "Hi abc")` => `True`
- `ends_with_other("HELLO", "hello")` => `True`
- `ends_with_other("HELLO WORLD", "hello")` => `False`
- `ends_with_other("abc", "def")` => `False`

Source: Coding Bat, http://codingbat.com/prob/p174314
a) Write a function `multiples_of` that consumes a list of natural numbers (called `numbers`) and a positive natural number (called `n`), and returns a (new) list containing all entries in `numbers` which are multiples of `n`. The new list must be in the same relative order as `numbers`, and the original list should be unchanged. Use recursion or abstract list functions.

For example:

Constructing a new list:

- `multiples_of([], 4) => []`
- `multiples_of([18, 5, 19, 21, 300, 0, 4], 3) => [18, 21, 300, 0]

❖ Note: The list that is consumed should remain the same.
b) Write a function `modify_multiples` that consumes a list of natural numbers (called `numbers`) and a positive natural number (called `n`), and mutates `numbers` so that all multiples of `n` are set to 0. The function returns `None`.

For example:

**Mutating numbers:**

```python
# if nums = [], after calling
# modify_multiples(nums,4), nums is []
# if nums = [18, 5, 19, 21, 300, 0, 4], after
# calling modify_multiples(nums, 3),
# nums is [0, 5, 19, 0, 0, 0, 4]
```
QUESTION 4 (MODULE 5 – ACCUMULATIVE RECURSION)

• Write an accumulatively recursive function `find_all` that consumes a list of strings `lst` and a string `s`, and returns the list of indices of positions in `lst` with string `s`. Recall that the first position in a list has index 0.

• For example,
  - `find_all(["a", "v", "d", "v"], "v") => [1,3]`
  - `find_all(["a", "v", "d", "v"], "q") => []`
QUESTION 5 (MODULE 5 – GENERATIVE RECURSION)

- Write an generative recursive function `find_all` that consumes a list of strings `lst` and a string `s`, and returns the list of indices of positions in `lst` with string `s`. Recall that the first position in a list has index 0.

- For example,
  - `find_all(["a", "v", "d", "v"], "v")` => `[1, 3]`
  - `find_all(["a", "v", "d", "v"], "q")` => `[]`
STUDY TIPS

• Review strategies:
  – Spaced practicing
  – Make own review notes
  – Good Sleep and Rest
  – Ask questions
  – Teach your friends
  – Come to office hours

• Review materials:
  – Course notes
  – Assignments
  – Tutorial Problems
  – Module Practices
  – Style Guide