CS61A Exam Prep 13

spring 2021 Final Q7

7. (2.0 points) Don't Be Late For Class!

All of the EECS class description pages list the days and times for the class in a particular format. For example:

```
MoWe 10:00AM - 10:59AM
```

Write a regular expression that matches that date/time format and *does* not match similar formats. Consult the doctests for examples of what your regular expression should and should not match. All possible combinations of days should be matched.

Remember that you can use regexr.com to try out regular expressions and use code.cs61a.org to run the doctests.

Note below that the second and third blanks will be filled by the same expression (b).

a) (Mo|Tu|We|Th|Fr)+

b) ((2[0-3])|(1?\d)):[0-5]\d (AM|PM)

```
import re

def find_time(class_descrip):
    """
    >>> find_time("(Fall 2021): MoWe 10:00AM - 10:59AM, Online - Hug")
    'MoWe 10:00AM - 10:59AM'
    >>> find_time("(Fall 2021): MoWeFr 10:00AM - 10:59AM, Lewis 100 - Nicholas Weaver")
    'MoWeFr 10:00AM - 10:59AM'
    >>> find_time("(Spring 2021): TuTh 12:30PM - 1:59PM, Online - Rao, Parekh")
    'TuTh 12:30PM - 1:59PM'
    >>> find_time("The class that never happens! TuTh 26:62PM - 1:62PM, The QuantumVerse")
    >>> find_time("Meet me on the glade from 2:00PM - 2:59PM today for a mochi muffin!")
    >>> find_time("My calendar is free TuTh 11-12 PM, how about yours?")
    >>> find_time("I wanna find a class for Fall 2021 that's @ Tuesday 9:00, seen any?")
    """
    match = re.search(r"_________\s_________ - _________", class_descrip)
    #                    (a)      (b)       (b)
    return match and match.group(0)
```

spring 2021 Final Q8a

```
start: comp

?comp: "[" expression "for" IDENTIFIER "in" IDENTIFIER "]"

expression: IDENTIFIER operation*

operation: OPERATOR NUMBER

IDENTIFIER: /[a-zA-Z]+/

OPERATOR: "*" | "/" | "+" | "-"

%import common.NUMBER
%ignore /\s+/
```

(a) (1.0 pt) Select all of the non-terminal symbols in the grammar:

- [x] comp
- [x] expression
- [x] operation
- [ ] NUMBER
- [ ] IDENTIFIER
- [ ] OPERATOR

spring 2021 Final Q8b

```
start: comp

?comp: "[" expression "for" IDENTIFIER "in" IDENTIFIER "]"

expression: IDENTIFIER operation*

operation: OPERATOR NUMBER

IDENTIFIER: /[a-zA-Z]+/

OPERATOR: "*" | "/" | "+" | "-"

%import common.NUMBER
%ignore /\s+/
```

(b) (1.0 pt) Which of the following comprehensions would be successfully parsed by the grammar?

- [x] `[ x * 2 for x in list ]`
- [x] `[ x for x in list ]`
- [ ] `[ x ** 2 for x in list ]`
- [ ] `[ x + 2 for x in list if x == 1 ]`
- [ ] `[ x * y for x in list for y in list2 ]`
- [ ] `[ x - 2 for x in my_list ]`
- [ ] `[ x - y for (x,y) in tuples ]`

spring 2021 Final Q8c

```
start: comp

?comp: "[" expression "for" IDENTIFIER "in" IDENTIFIER "]"

expression: IDENTIFIER operation*

operation: OPERATOR NUMBER

IDENTIFIER: /[a-zA-Z]+/

OPERATOR: "*" | "/" | "+" | "-"

%import common.NUMBER
%ignore /\s+/
```

(c) (1.0 pt) Which line would we need to modify to add support for a `%` operator, like in the expression `[ n % 2 for n in numbers ]`?

- (●) `OPERATOR: "*" | "/" | "+" | "-"` | "%"
- ( ) `IDENTIFIER: /[a-zA-z]+/`
- ( ) `operation: OPERATOR NUMBER`
- ( ) `expression: IDENTIFIER operation*`
- ( ) `?comp: "[" expression "for" IDENTIFIER "in" IDENTIFIER "]"`

Fall 2016 Midterm 2 Q7a

(a) (6 pt) Implement `sums`, which takes two positive integers `n` and `k`. It returns a list of lists containing all the ways that a list of `k` positive integers can sum to `n`. Results can appear in any order.

```
def sums(n, k):
    """Return the ways in which K positive integers can sum to N.

    >>> sums(2, 2)
    [[1, 1]]
    >>> sums(2, 3)
    []
    >>> sums(4, 2)
    [[3, 1], [2, 2], [1, 3]]
    >>> sums(5, 3)
    [[3, 1, 1], [2, 2, 1], [1, 3, 1], [2, 1, 2], [1, 2, 2], [1, 1, 3]]
    """
    if _____:
        return _____

    y = []

    for x in _____:
        y.extend([_____ for s in sums(_____)])

    return y
```

n=4
k=2

k == 1

[[n]]

range(1, n)

s + [x]

n-x, k-1

sums(3, 1)
[[3]]

[3, 1]