Regular Expressions

CSE Cracks – 20XX
December 30, 2023

This quiz is open note, open book, and open world. Assume all regular expressions are done in Python using the `re` standard library. Good luck!

Question:	1	2	3	4	5	6
Points:	5	20	5	0	10	5
Score:						
Question:	7	8	9	10		Total
Points:	5	10	20	20		100
Score:						

Name: _____

Email: _____

I acknowledge that I have neither given nor received inappropriate help on this exam, and have abided by the letter and spirit of the University of California, Santa Cruz Code of Academic Integrity while taking this exam.

Signature: _____

1. (5 points) Regular expressions are implemented as either a core feature or in the standard library of almost every major programming language.

   True / False

2. (20 points) Match the following terms to their corresponding definitions.

_____ Character Class	A. A set of character where any single member of the group can be matched.
_____ Anchor	B. A special character that can be used to match the beginning or end of a line.
_____ Word Boundary	C. The empty string between (`[\W^]` and `\w`) or between (`\w` and `[\W$]`).
_____ Kleene Star	D. A repetition operator that matches the range $[0, infinity]$.
_____ Group	E. A collection of character that can be treated as a single unit.
_____ Disjunction	F. An operator that allows us to select one of two options.
_____ Back Reference	G. A special character that allows us to invoke a previous group.
	H. The set of all alphanumeric characters and underscore.
	I. All digits.
	J. A repetition operator that matches the range $[1, infinity]$.
	K. An operator that allows us to select both of two options.

3. (5 points) Which of the following regular expressions would be best to match a 10-digit phone number formatted as: '123 456-7890'. (Assume any stretch of continuous whitespace is a single space character.)

   A. `r'\d{3} \d{3}-\d{4}'`

   B. `r'\d{10}'`

   C. `r'\d* \d*-\d*'`

   D. `r'\d+ \d+-\d+'`

4. (0 points) Below is the opening paragraph (which is actually just one sentence) from *A Tale Of Two Cities* written by Charles Dickens. Future questions may reference this passage as "the provided passage".

   "It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way — in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only."

5. (10 points) In the provided passage, how many non-specific time periods are mentioned, i.e., how many matches are there for the following regular expression:

```
r'(age|season|epoch)\s+of\s+(\w+)'
```

6. (5 points) For each scenario, select the quantifier that is most appropriate.

   You want to match the leading zeros for some number. E.g., "00" for "005".
   <PART1>

   You want to match the negative sign for some number. E.g., "-" for "-9".
   <PART2>

   You want to match the main digits (before any decimal point) for a required number. E.g. "123" for "123".
   <PART3>

   (a) PART1:
   A. `?` B. `*` C. `+`

   (b) PART2:
   A. `?` B. `*` C. `+`

   (c) PART3:
   A. `?` B. `*` C. `+`

7. (5 points) Which of the following does the regex `r'I'm So{3,4} Hungry!'` match? Select all that apply.

   A. I'm So Hungry!

   B. I'm Soo Hungry!

   C. I'm Sooo Hungry!

   D. I'm Soooo Hungry!

8. (10 points) Suppose that we are trying to write a script extract name information from text and put it into a CSV (comma-separated value) file. The order of the columns in our CSV file are: first name, last name, and title. As part of our script, we have a regular expression that looks for people that have their name's written as "last, first".

```
import re

def create_csv_line(text_line):
    regex = r'^\s*((Dr).?)?\s*([^,]+)\s*,\s*(.+)\s*$'
    replacement = MY_REPLACEMENT_STRING

    return re.sub(regex, replacement, text_line)
```

Fill in the blanks in `MY_REPLACEMENT_STRING` to make the above code work correctly.
`MY_REPLACEMENT_STRING = r'`<BLANK1>,<BLANK2>,<BLANK3>`'`

A. BLANK1: _____

B. BLANK2: _____

C. BLANK3: _____

9. (20 points) Create a regular expression that matches successfully completes a game a golf with the table below.
   Specifics:

   - Match all values in the `Match` column.
   - Do not match any values in the `No Match` column.
   - Write you regex as a raw string using a single or double quotes (not triple quotes).
   - Treat the contents of each table cell as a string (so you do not have the match the quotes).
   - You may assume that any contiguous whitespace is a single space character.
   - You only need to match (or not match) the values in the table, you do not need to extend this pattern to unseen values.

Match	No Match
`'12:00 AM'`	`'00:00'`
`'05:30 PM'`	`'17:30'`
`'01:45 AM'`	`'01:65 AM'`
`'10:10 PM'`	`'10:10 ZZ'`
`'12:34 PM'`	`'12:34 pm'`
`'11:59 PM'`	`'23:59'`
	`'123:45 AM'`
	`'12:345 PM'`

10. (20 points) Implement a function with the folowing signature and description:

```
import re

def compute(text):
    """
    Compute the result of the binary expression represented in the |text| variable.
    The possible operators are: "+", "-", "*", and "/".
    Operands may be any real number.
    The the operation is division, the RHS (denominator) will not be zero.
    """

    return NotImplemented
```

Create a regular expression that matches successfully completes a game a golf with the table below. Specifics:

- Your function must use regular expressions.
- You may not use `eval()` or any other Python ast functionality.
- You may only import modules from the Python standard library.
- You should return a float that is the result of the binary operation represented by `text`.
- The operator will be one of: $\{+, -, *, /\}$.
- Operands may be any real number.