Name:

Student ID:

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!

By signing below, 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.

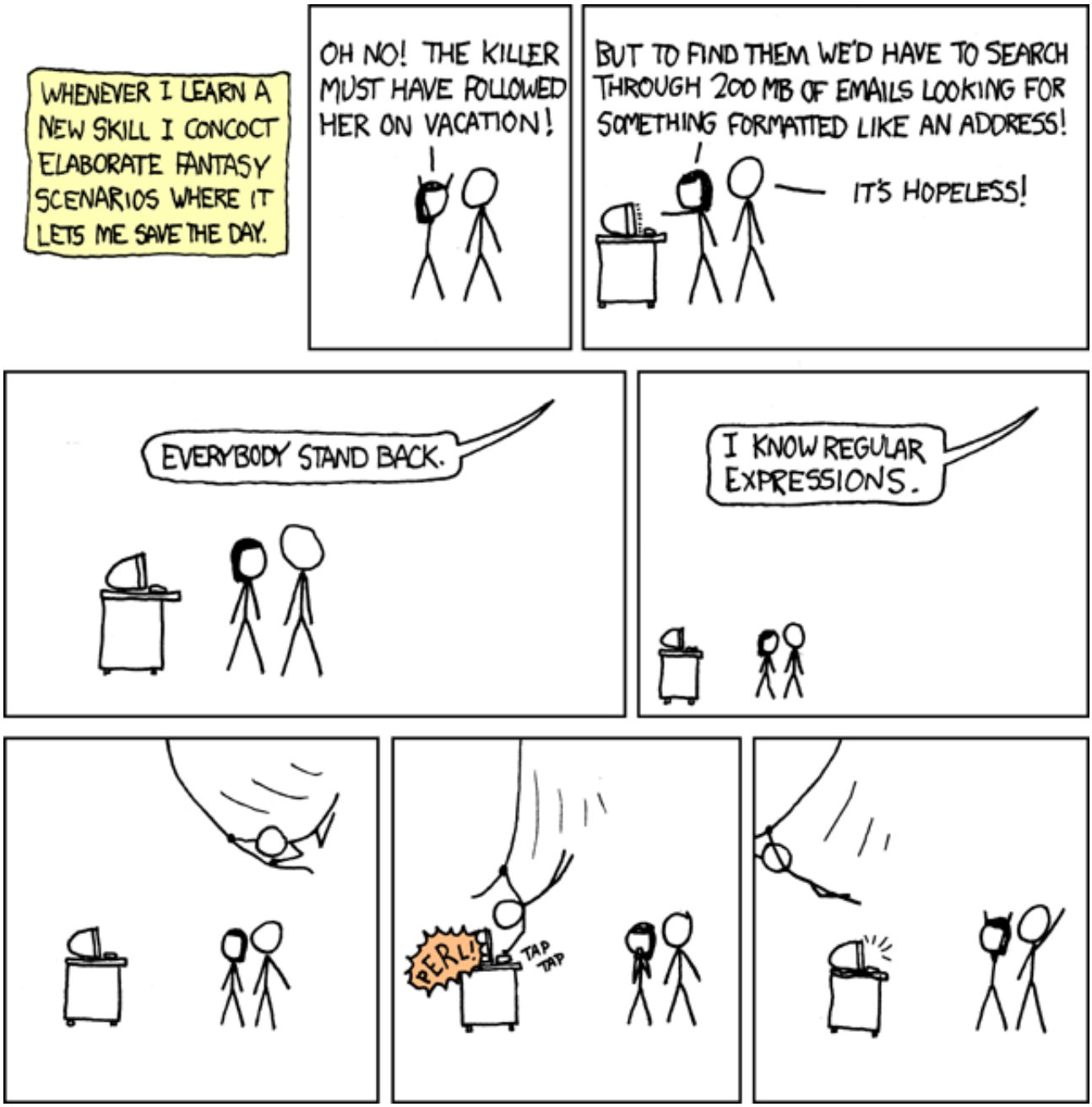
Signature:

Fill in the box below if you believe your exam may require manual grading, e.g., you crossed out answers or needed space outside a designated answer box.

☐ My exam may require manual grading.

Question 1 [Ice Breaker] (5 Points)

Taking inspiration from the XKCD comic below, how would you save the day using regular expressions?



Place your answer within the boxed region below.

Question 2 [Regular Expression in Programming Languages] (5 Points)

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

Fill in the circle that corresponds to your answer.

◯ True

◯ False

Question 3 [Regular Expression Vocabulary] (20 Points)

Match the following terms to their corresponding definitions.

Fill in the each box on the left with a letter from the right.

☐ Back Reference

☐ Anchor

☐ Kleene Star

☐ Disjunction

☐ Character Class

☐ Group

☐ Word Boundary

A. An operator that allows us to select both of two options.

B. A special character that allows us to invoke a previous group.

C. The empty string between (`[\W^]` and `\w`) or between (`\w` and `[\W$]`).

D. A collection of character that can be treated as a single unit.

E. A special character that can be used to match the beginning or end of a line.

F. The set of all alphanumeric characters and underscore.

G. A repetition operator that matches the range $[1, infinity]$.

H. All digits.

I. A set of character where any single member of the group can be matched.

J. An operator that allows us to select one of two options.

K. A repetition operator that matches the range $[0, infinity]$.

Question 4 [Basic Regular Expressions] (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.)

Fill in the circle that corresponds to your answer.

- ◯ `r'\d* \d*-\d*'`
- ◯ `r'\d{3} \d{3}-\d{4}'`
- ◯ `r'\d+ \d+-\d+'`
- ◯ `r'\d{10}'`

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."

Question 5 [Passage Search] (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+)'
```

Place your answer within the boxed region below.

Question 6 [Quantifiers] (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>

For each part (denoted by angle brackets), fill in the circle that corresponds to your answer.

PART1:

◯ `*` ◯ `?` ◯ `+`

PART2:

◯ `+` ◯ `*` ◯ `?`

PART3:

◯ `+` ◯ `?` ◯ `*`

Question 7 [General Quantification] (5 Points)

Which of the following does the regex `r'Lo{2,3}ng Cat'` match? Select all that apply.

Fill in all boxes that corresponds to your answers.

☐ Loong Cat

☐ Loooong Cat

☐ Long Cat

☐ Looong Cat

Question 8 [Backreference Matching] (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'<A>,<B>,<C>'`

For each part (denoted by angle brackets), place your answer in the associated box.

A: ☐ B: ☐ C: ☐

Question 9 [Regex Golf] (15 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'

Place your answer within the boxed region below.

Question 10 [Write a Function] (20 Points)

Implement a function with the following 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.
    If the operation is division, the RHS (denominator) will not be zero.
    """

    return NotImplemented
```

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.

Place your answer within the boxed region on the next page.