CSC148 - Object-Oriented Programming

1. Below is the initializer from an incorrect implementation of the `Spinner` class from this week's prep:

```
class Spinner:
    slots: int
    position: int

    def __init__(self, size: int) -> None:
        """Initialize a new spinner with <size> slots.

        A spinner's position always starts at 0.
        """
        slots = size
        position = 0
```

Looking at the results of the class doctests, we observe that this code raises an error:

```
>>> s = Spinner(8)
>>> s.position
ERROR ...
```

(i) On a separate piece of paper, draw a memory model diagram for this code. (see pg 3)

(ii) What does this initializer actually do?

creates 2 local variables

(iii) What error is raised when we run this doctest (i.e., be more specific than just `ERROR ...`).

Attribute error: Spinner class has no attribute position

2. Here is the documentation for the `Tweet` class you saw in Prep2, with one new method called `edit` added:

```
class Tweet:
    """A tweet, like in Twitter.

    === Attributes ===
    content: the contents of the tweet.
    userid: the id of the user who wrote the tweet.
    created_at: the date the tweet was written.
    likes: the number of likes this tweet has received.
    """
    content: str
    userid: str
    created_at: date
    likes: int

    def __init__(self, who: str, when: date, what: str) -> None:
        """Initialize a new Tweet."""

    def edit(self, new_content: str) -> None:
        """Replace the contents of this tweet with the new message.

        >>> t = Tweet('Rukhsana', date(2017, 9, 16), 'Hey!')
        >>> t.edit('Rukhsana is cool')
        >>> t.content
        'Rukhsana is cool'
        """
```

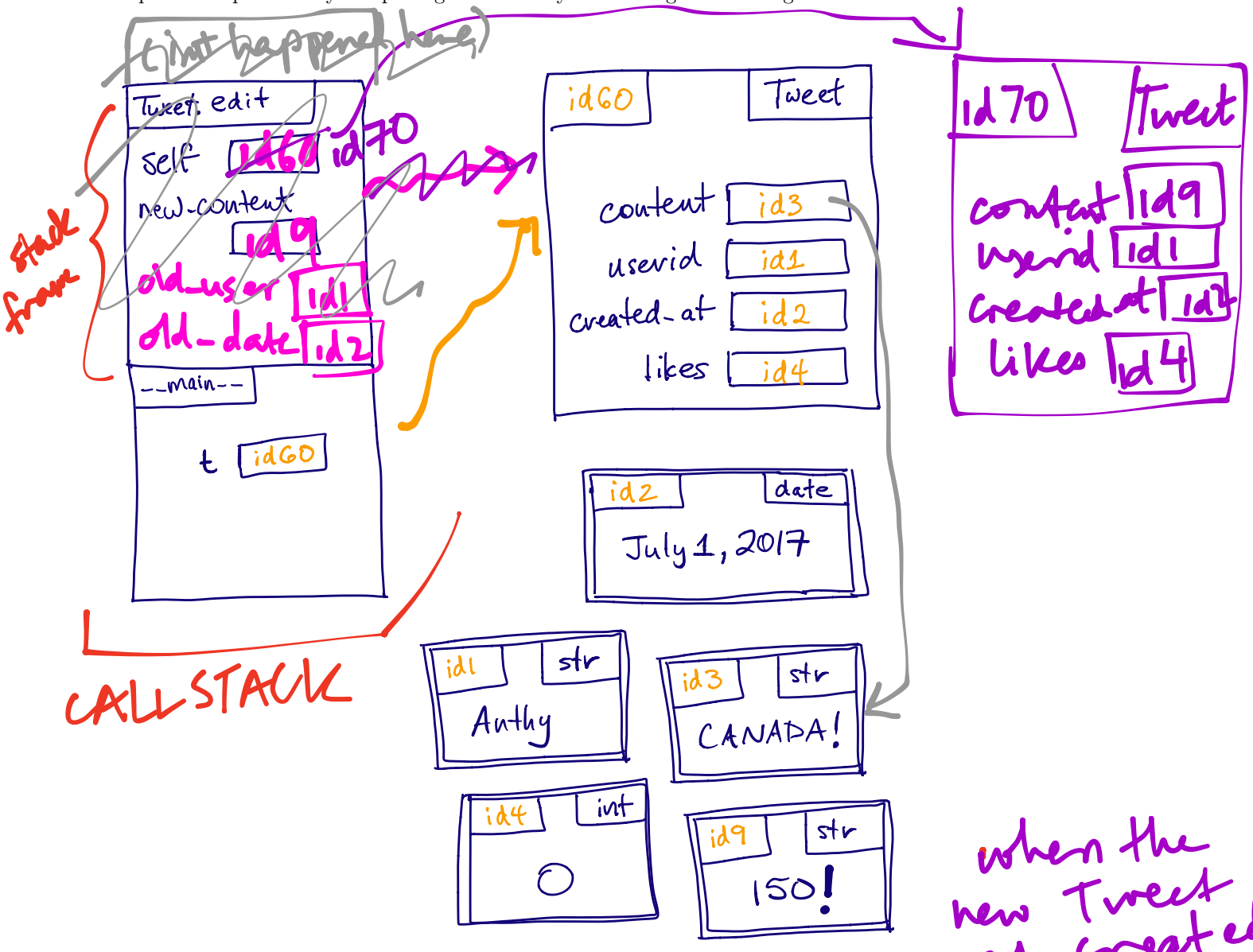
3. Here's an incorrect implementation of `edit`:

```
def edit(self, new_content: str) -> None:
    old_user = self.userid
    old_date = self.created_at
    self = Tweet(old_user, old_date, new_content)
```

When we run the following code, the wrong thing is printed:

```
>>> t = Tweet('Anthy', date(2017, 7, 1), 'CANADA!')
>>> t.edit('150!')
>>> print(t.content)  # Prints 'CANADA!', not '150!'
```

Explain this problem by completing this memory model diagram showing the call to `edit`.



4. Implement method `edit` correctly in the space below.

```
def edit(self, new_content: str) -> None:
```

self.content = new_content

when the new Tweet was created, __init__ was added to call stack:

CSC148 - Object-Oriented Programming

1. Below is the initializer from an incorrect implementation of the `Spinner` class from this week's prep:

```
class Spinner:
    slots: int
    position: int

    def __init__(self, size: int) -> None:
        """Initialize a new spinner with <size> slots.

        A spinner's position always starts at 0.
        """
        slots = size
        position = 0
```

correct if wrong

self. self.

1. Create the instance
2. Call __init__ + send id of new instance to self

Looking at the results of the class doctests, we observe that this code raises an error:

```
>>> s = Spinner(8)
>>> s.position
ERROR ...
```

(i) On a separate piece of paper, draw a memory model diagram for this code.