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

```python
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:

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

Explain two things:

(i) What the initializer's implementation actually does.
(ii) What error is raised when we run this doctest (i.e., be more specific than just ERROR ...).

2. Here is the documentation for the Tweet class, with one new method edit added:

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

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

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

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

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

4. Implement method edit correctly in the space below.

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