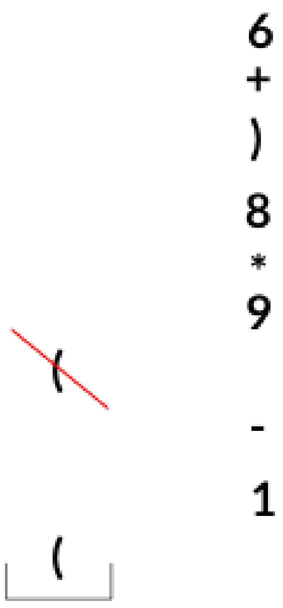
CSC148 Worksheet 9 Solution

Hyungmo Gu

April 21, 2020

Question 1

Expression 1:

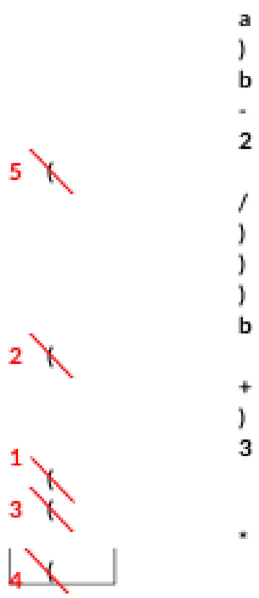


Stack

Were the parentheses balanced?

Yes [✓ No]

Expression 2:

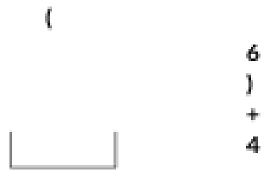


Stack

Were the parentheses balanced?

[✓ Yes] No

Expression 3 :



Stack

Were the parentheses balanced?

Yes [✓ No]

Question 2

a. For each character being received,

1. If the character is left parenthesis, then we need to store it in stack using *push()* method
2. If the character is right parenthesis
   1. First, check for the non-emptiness of stack.
   2. If the list is not empty, then we need to pop an element form stack.
   3. If list is empty, then we need to raise error.
3. If the character is other than left or right parenthesis, then pass the character.

b. We will know the parenthesis are balanced when the number of elements in stack is zero after traversing a string.

Question 3

```python
    def is_balanced(line: str) -> bool:
        """Return whether <line> contains balanced parentheses.

        Ignore square and curly brackets.

        >>> is_balanced('(a * (3 + b))')
        True
        >>> is_balanced('(a * (3 + b]]') # Note that the two ']'s don't matter
        False
        >>> is_balanced('1 + 2(x-y)}') # Note that the '}' doesn't matter
        True
        >>> is_balanced('3 - (x')
        False
        """
        parenthesis_stack = Stack()

        for character in line:
            # If the character is left parenthesis,
            if character == '(':
                # Store it in stack
                parenthesis_stack.push('(')
            # If the character is right parenthesis,
            elif character == ')':
                # Check for the non-emptiness of stack.
                if parenthesis_stack.is_empty():
                    # if empty, return false.
                    return False

                # If the list is not empty, then pop an element form stack.
                parenthesis_stack.pop()

        # Check parenthesis are balanced by checking stack is empty.
        if not parenthesis_stack.is_empty():
            return False

        return True
```

Listing 1: worksheet_9_q3_solution.py

Question 4

```
def is_balanced(line: str) -> bool:
    """Return whether <line> contains balanced parentheses.

    >>> is_balanced('abc')
    True
    >>> is_balanced('(a * (3 + b))')
    True
    >>> is_balanced('(a * (3 + b]]')
    False
    >>> is_balanced('(a * [3 + b])')
    True
    >>> is_balanced('1 + 2(x-y)}')
    False
    >>> is_balanced('{3 + [2 * 4(x-y)]}')
    True
    >>> is_balanced('3 - (x')
```

```python
        False
        """
        brackets_stack = Stack()

        for character in line:
            # If the character is one of '[', '{'. or '(',
            if (character == '(' or
                character == '[' or
                character == '{'):
                # Store it in stack
                brackets_stack.push(character)
            # If the character is one of ']', '}', or ')',
            elif (character == ')' or
                character == ']' or
                character == '}'):
                # Check for the non-emptiness of stack.
                if brackets_stack.is_empty():
                    # if empty, return false.
                    return False

                # If the list is not empty, then pop an element form stack
    .
                left_bracket = brackets_stack.pop()

                # If popped bracket doesn't match, then return false
                if ((left_bracket == '(' and character != ')') or
                    (left_bracket == '[' and character != ']') or
                    (left_bracket == '{' and character != '}')):

                    return False


        # Check parenthesis are balanced by checking stack is empty.
        if not brackets_stack.is_empty():
            return False

        return True
```
Listing 2: worksheet_9_q4_solution.py