FIT1008 – Intro to Computer Science
Solutions for Tutorial 7

Semester 1, 2018

Exercise 1

```
from my_stack import Stack


def is_matched(expression):
    left_bracket = "({["
    right_bracket = ")}]"
    stack = Stack(len(expression))
    for character in expression:
        if character in left_bracket:
            stack.push(character)
        elif character in right_bracket:
            #  it wont be matched if the stack is empty
            if stack.is_empty():
                return False
            # it wont be matched if the character
            # I pop is not the equivalent on the left
            if right_bracket.index(character) !=
                     left_bracket.index(stack.pop()):
                return False
    return stack.is_empty()


def main():
    expression = input("Enter␣expression:␣")
    if is_matched(expression):
        print("Correct␣expression")
    else:
        print("Incorrect␣expression")


if __name__ == "__main__":
    main()
```

Exercise 2

```
def index(self, item):
     low = 0
     high = len(self)-1

     while low <= high:
        mid = (low + high)//2

        if item == self.the_array[mid]: # found item
            if low == high: # found first item
                return low
            high = mid
        elif item < self.the_array[mid]:
            high = mid - 1
        else
            low = mid + 1

    raise ValueError(str(item) + "␣not␣in␣the␣list")
```

Exercise 3

```
def reverse(my_queue):

    my_stack = Stack(my_queue.size())     # used to reverse
    result_q = Queue(my_queue.size())     # used for computing the result

    while not my_queue.is_empty():
        elem = my_queue.serve()
        my_stack.push(elem)
        result_q.append(elem)

    while not my_stack.is_empty():
        my_queue.append(result_q.serve())
        item = my_stack.pop()
        if item:                 # empty string is False in boolean context
            result_q.append(item)

    return result_q
```

Exercise 4

```
def print_reverse_queue(self):
    idx = self.rear
    for _ in range(self.count)):
        print(self.the_array[idx])
        idx = (idx - 1) % len(self.the_array)
```