Rahul Vaidun
rvaidun@ucsc.edu
1 April 2021

CSE 13S Spring 2021
Assignment 3: Sorting: Putting your affairs in order
Design Document

I. Prelab

Part 1:

1. **How many rounds of swapping will be needed to sort the numbers?**
   8,22,7,9,31,5,13 in ascending order using Bubble Sort
   Round 1: 8, 7, 9, 22, 5, 13, 31
   Round 2: 7, 8, 9, 5, 13, 22, 31
   Round 3: 7, 8, 5, 9, 13, 22, 31
   Round 4: 7, 5, 8, 9, 13, 22, 31
   Round 5: 5, 8, 8, 9, 13, 22, 31
   It will take 5 rounds of swapping
2. **How many comparisons can we expect to see in the worse case scenario for Bubble Sort?**
   In the worst case scenario there will be n^2comparisons where n is the total amount of items in the array

Part 2:

1. **The worst time complexity for Shell Sort depends on the sequence of gaps. Investigate why this is the case. How can you improve the time complexity of this sort by changing the gap size?**
   The worst case time complexity for Shell Sort depends on the sequence of gaps because the higher the initial gap the longer it takes to reduce. The gap between the items continually reduces but if you start with a higher gap it will take longer to reduce than if you start with a lower gap

Part 3:

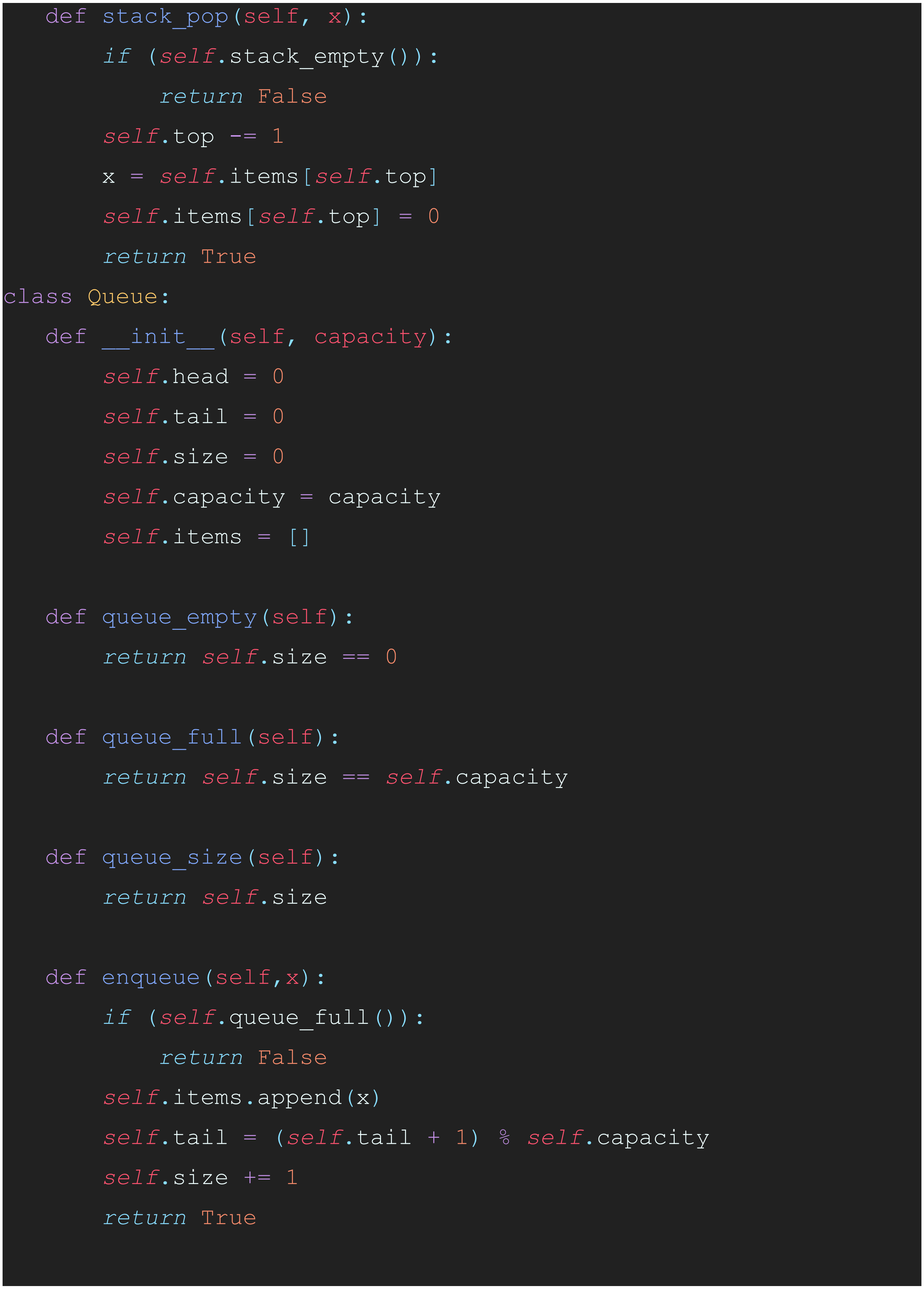
1. **Quicksort, with a worst case time complexity of O(n^2) doesn't live up to its name. Investigate and explain why Quicksort isn't doomed by its worst case scenario.**
   Quicksort isn't doomed by its worst case scenario because by changing the algorithm used to find the pivot we can reduce quicksort time complexity. Using random pivoting helps reduces the expected time complexity to $O(n \log n)$
   I used GeeksforGeeks with help on this problem
   https://www.geeksforgeeks.org/quicksort-using-random-pivoting/

Part 4:

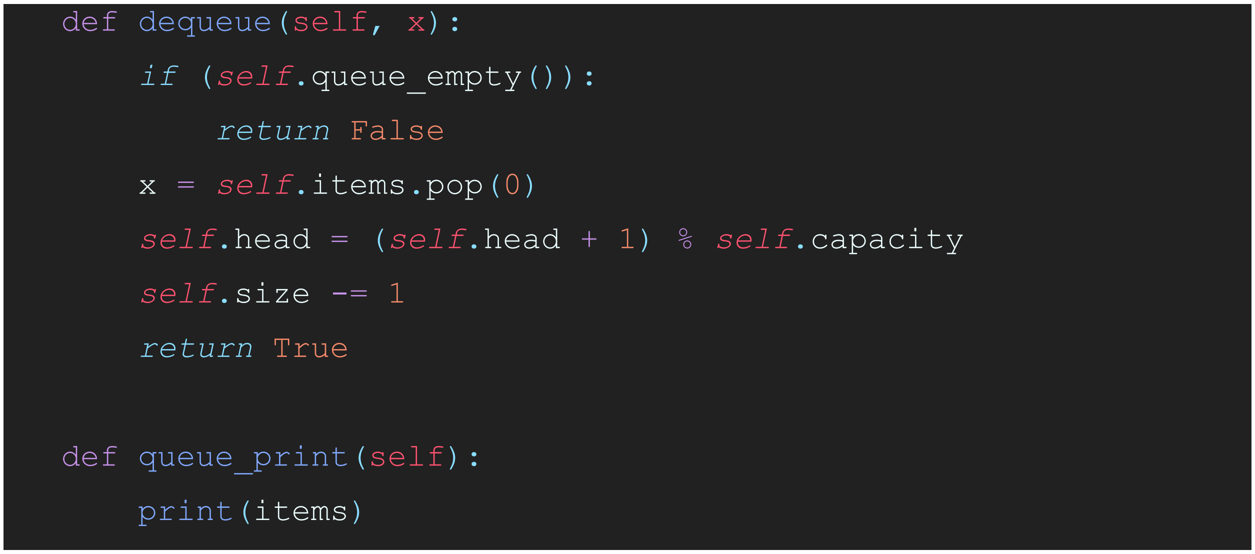
1. **Explain how you plan on keeping track of the number of moves and comparisons since each sort will reside in its own file**
   I will keep track of the number of moves and comparisons by adding an extra header file to keep track of these integers. I can then utilize the extern keyword to reference the variables outside of the file

II. Pseudocode

```
class Stack:
 def __init__(self, capacity):
   self.capacity = capacity
   self.top = 0
   self.items = [];

   def stack_empty(self):
       return self.top == 0;

   def stack_print(self):
       print(self.items)

   def stack_full(self):
       return self.top == self.capacity

   def stack_size(self):
       return self.top

   def stack_push(self, x):
       if (self.stack_full()):
           return False
       self.items[self.top] = x
       self.top += 1
       return True
```



```python
   def stack_pop(self, x):
       if (self.stack_empty()):
           return False
       self.top -= 1
       x = self.items[self.top]
       self.items[self.top] = 0
       return True
class Queue:
   def __init__(self, capacity):
       self.head = 0
       self.tail = 0
       self.size = 0
       self.capacity = capacity
       self.items = []

   def queue_empty(self):
       return self.size == 0

   def queue_full(self):
       return self.size == self.capacity

   def queue_size(self):
       return self.size

   def enqueue(self,x):
       if (self.queue_full()):
           return False
       self.items.append(x)
       self.tail = (self.tail + 1) % self.capacity
       self.size += 1
       return True
```



```python
    def dequeue(self, x):
        if (self.queue_empty()):
            return False
        x = self.items.pop(0)
        self.head = (self.head + 1) % self.capacity
        self.size -= 1
        return True

    def queue_print(self):
        print(items)
```