FIT1008 – Intro to Computer Science
Tutorial 9

Semester 1, 2017

Objectives of this tutorial

- To understand how queues work.
- To be able to implement methods for the queue implemented as a circular queue.
- To understand how linked structures, in particular stacks, work.

Exercise 1

Consider a `Stack` ADT that implements a stack of strings using some data structure (you do not need to know which one) and defines the usual methods, where n is the size of the stack:

```
Stack(n)
pop()
push(item)
size()
is_empty()
```

Consider a `Queue` ADT that implements a queue of strings using some data structure (you do not need to know which one) and defines the usual methods, where n is the size of the queue:

```
Queue(n)
serve()
append(item)
size()
is_empty()
```

Use stack and queue operations to define the function

`reverse(my_queue)`

which takes a queue of strings called `my_queue`, returns a new one containing all non-empty strings from `my_queue` in reverse order, and does this by using a stack. Note that, at the end of the method, `my_queue` must contain the same elements as when it started, and in the same order (i.e., if you need to modify `my_queue`, make sure you leave it as it was).
For example, if `my_queue` has the following 5 elements :

`"Hello", "Goodbye", "Not now", "", "Later"`

where "Hello" is the item at the front, then the method will return the following queue, which has 4 elements with "Later" at the front:

`"Later", "Not now", "Goodbye","Hello"`

Exercise 2

Consider the circular implementation of the `Queue` ADT given in the Lecture notes and the following piece of code. Show the contents of `my_queue` after each `append` and `serve` (but show only those items in the array that correspond to items in the queue).

```
n = 10

my_queue = Queue(5)
while not n == 0 and not is_full(my_queue):
    my_queue.append(n)
    n = n // 2
while not my_queue.is_empty():
    x = my_queue.serve()
    x = x - 1
    my_queue.append(x)
    if x < 5:
        break
```

What is the value of x at the end of the computation?

Exercise 3

Assume class Queue implements a queue as a circular queue as in lectures. Write a Python method, *print_reverse_queue(self)*, for the class Queue, which prints all the items in the queue from rear to front, without changing the queue.

Exercise 4

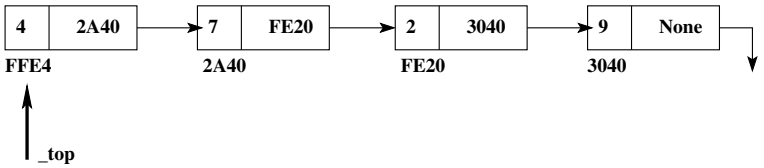
Consider the `Stack` class given in the lecture notes for a linked stack. Consider adding a method `sum_all` to the class whose function is to return the sum of all elements in the stack (zero if the stack is empty) without modifying the stack itself. The following shows three possible implementations of such method:

(i)
```
def sum_all(self):
    sum = 0
    while (not self.is_empty()):
        sum += self._top.item
        self._top = self._top.next
    return sum
```

(ii)
```
def sum_all(self):
    current = self._top
    sum = 0
    while current.next != None:
        sum += current.item
        current = current.next
    return sum
```

(iii)
```
def sum_all(self):
    current = self._top
    sum = 0
    while current != None:
        current = current.next
        sum += current.item
    return sum
```

Consider an object `my_stack` with the form



Show the effect on the stack of calling `result = my_stack.sum_all()` for each definition above. Show also the final value of `result`. Provide a correct definition for the above method.