FIT1008 – Intro to Computer Science
Tutorial 7

Semester 1, 2018

Objectives of this tutorial

- To understand how stacks work and how can they be used in practical problems.
- To be able to work with a SortedList ADT.

Exercise 1

- A mathematical expression is provided in a string, which may contain opening and closing parenthesis. Write a python function to determine if the parenthesis are balanced. **Hint:** This is easy if you use a Stack.
- Extend your function to include checks for balanced strings including, also curly and square brackets.

Exercise 2

Assume the class SortedList is an array implementation of the Sorted List ADT, as given in lectures. Write a method *index(self, item)* for SortedList which has a worst time complexity of *O(log(N))*, where N is the length of the list. The method *index* finds the first index of *item* in the list, and raises a *valueError* if the item is not in the list.

Exercise 3

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 4

Study the implementation below, which uses an array to implement a Queue. As opposed to the linear queue covered in the lectures, this implementation does not waste space.

```
1
2  class CircularQueue:
3
4      def __init__(self, size):
5          assert size > 0, "Size must be positive"
6          self.array = [None] * size
7          self.reset()
8
9      def reset(self):
10         self.front = 0
11         self.rear = 0
12         self.count = 0
13
14     def is_empty(self):
15         return self.count == 0
16
17     def is_full(self):
18         return self.count >= len(self.array)
19
20     def serve(self):
21         assert self.count > 0, "Empty queue"
22         item = self.array[self.front]
23         self.front = (self.front + 1) % len(self.array)
24         self.count -= 1
25         return item
26
27     def append(self, item):
28         assert not self.is_full(), "Full queue"
29         self.array[self.rear] = item
30         self.rear = (self.rear + 1) % len(self.array)
31         self.count += 1
```

Write a Python method, *print_reverse_queue(self)*, for the class `CircularQueue`, which prints all the items in the queue from rear to front (without changing the queue).