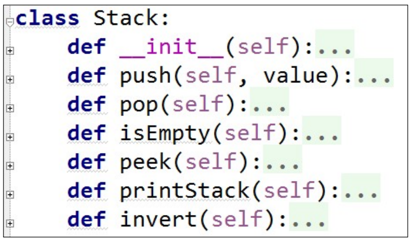
INF1008: Data Structures and Algorithms

Tutorial 2: Basic Data Structures

A partial implementation of all ADTs used in this tutorial (Stack, Queue, SinglyLinkedList) is provided in the lecture note 1 that can be downloaded from xSite.

Q1. Using the Stack class provided in the lecture note, write a method invert() to invert the contents of the stack. You may use additional stacks in your function. Note: If the number 3 is at the top of the stack s, after calling s.invert(), 3 will be at the bottom of s.

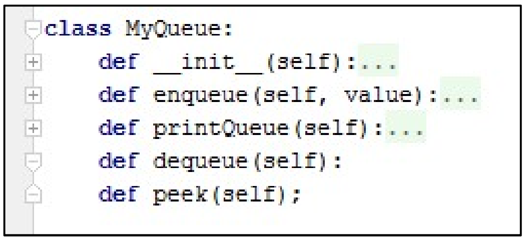


```
class Stack:
    def __init__(self):...
    def push(self, value):...
    def pop(self):...
    def isEmpty(self):...
    def peek(self):...
    def printStack(self):...
    def invert(self):...
```

Q2. Suppose that Q is a Queue. List the content of the queue after each operation and show the output value if a value is returned from the operation.

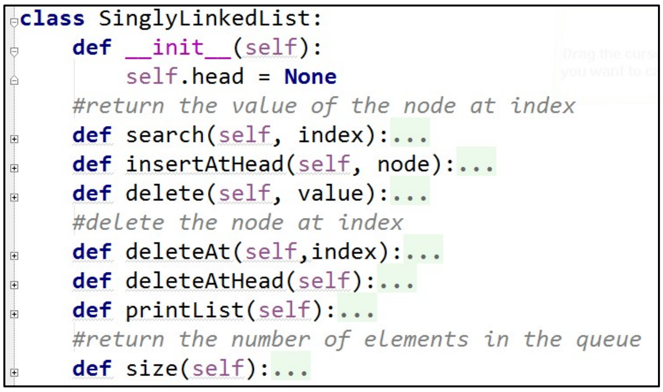
Operation	Output	[Front – Queue – Rear]
`Q = Queue()`		
`Q.isEmpty( )`		
`Q.enqueue(8)`		
`Q.enqueue(-5 )`		
`Q.dequeue( )`		
`Q.enqueue(2)`		
`Q.peek( )`		
`Q.dequeue( )`		
`Q.isEmpty( )`		
`Q.peek( )`		
`Q.dequeue( )`		

Q3. Implement the following Queue ADT using a Singly Linked List.



```
class MyQueue:
    def __init__(self):...
    def enqueue(self, value):...
    def printQueue(self):...
    def dequeue(self):
    def peek(self);
```

An implementation of the SinglyLinkedList ADT is provided in the lecture note. The API is as follow:



```
class SinglyLinkedList:
    def __init__(self):
        self.head = None
    #return the value of the node at index
    def search(self, index):...
    def insertAtHead(self, node):...
    def delete(self, value):...
    #delete the node at index
    def deleteAt(self,index):...
    def deleteAtHead(self):...
    def printList(self):...
    #return the number of elements in the queue
    def size(self):...
```

Q4. Write an algorithm to merge two Singly Linked Lists. Assume that the data in each list are in non-decreasing order. The algorithm should return a one Singly Linked List, containing all the data in non-decreasing order. You modify only the next fields of the nodes.

For example, if the lists are:

`3 6 6 10 45 45 50` and
`2 3 55 60`

After the merge, the result is:

`2 3 3 6 6 10 45 45 50 55 60`

Q5. Write a program that reads in a sequence of characters, and determines whether its parentheses, braces, and curly braces are "balanced." Your program should read one line of input containing what is supposed to be a properly-formed expression in algebra and tells whether it is in fact legal. The expression could have several sets of grouping symbols of various kinds, () and [] and {}. Your program needs to make sure that these grouping symbols match up properly.