ADT vs Interface

ADT: a general specification of a set of data items and operations performed on those data items.

Interface: A java specific class that defines nothing more than public abstract methods without a body. Classes that implements an interface must included all the methods defined in that interface.

Both include a collection of methods that can be performed on a data type. However, ADT is a more general concept, while an interface is a java specific tool that help the programmer to organize.

Extends vs Implements

A extends B:

- class A inherits all the methods and instance variables from class B.
- A = subclass. B = superclass
- Subclass can only extends one superclass

A implements B

- Class A must implement all the methods defined in interface B
- One class can imple

Method overriding vs Overloading

Method overloading: when different methods have the same name but different signatures (a combination of method name and method parameters)

Method overriding: the replacement of an inherited method with a different implementation in a subclass.

Primitive type vs Reference type

Primitive types:

- information is stored in the environment
- int, float, char, long, boolean, etc.

Reference type: i

- Information is stored in the heap memory
- String, Queue<E>, SimpleLinkedList<E>

Assume that Shape is an interface and Rectangle and Triangle are classes that implement Shape. Which of the following are valid statements?

(a) Rectangle myRectangle = new Rectangle();

(b) Shape mySecondRectangle = new Rectangle();

(c) Triangle myTriangle = new Shape();

(d) Triange mySecondTriangle = new Rectangle();

(e) Shape myShape = new Shape()

Assume that Shape is an interface and Rectangle and Triangle are classes that implement Shape. Which of the following are valid statements?

(a) Rectangle myRectangle = new Rectangle(); Valid

(b) Shape mySecondRectangle = new Rectangle(); Valid

(c) Triangle myTriangle = new Shape(); Invalid

(d) Triangle mySecondTriangle = new Rectangle(); Invalid

(e) Shape myShape = new Shape(); Invalid

Write the steps that an algorithm might take to efficiently check for valid parenthetical statements. (Modified from UIUC).

```
declare the stack variable
for each item in the String of brackets:
    if item is an opening bracket:
        push item to stack
    else if item is a closing bracket:
        pop an item from the stack
        If the popped item is NOT a matching pair for the closing bracket
            return false
if the stack is empty return True
else return False}
```

The runtime complexity for the radix sort you implemented in the lab is O(n*d) where n is the number of items to be sorted, while d is the number of digits of the largest item. Why is this?

For each item, the algorithm will:

Check a particular digit of the item (constant time). If the item is too small to contain the digit, then check the next item.

Place the item into the corresponding queues (constant time)

At the end of each loop, dequeue the items and return them to the original collection (also constant time).

The runtime for completing one iteration of the loop above is O(n). The number of times that the for loop is going to be executed is equal to d, the number of digits of the largest item. Thus, the runtime of radix sort is O(n*d).

Write a brief overview of how you might implement dequeue() for a queue class that uses a circular array?

```
If numItems == 0, then return null;

E toReturn = array[front]; //Grab the data at the front index

Front = (front + 1) % capacity; //

Decrement numItems

return toReturn
```