Homework 6 – Stacks, Queues

Part 1: Stacks

1. Override toString() for both implementations of StackADT, where the top value will be printed on the left.
   toString() do not change the internal state of the stack.

   In example:

```
StackADT<Integer> arrS = new AStack<Integer>(5);
arrS.push(1);
arrS.push(2);
arrS.push(3);
arrS.push(4);
arrS.pop();
System.out.println(arrS);
// Prints: 3 2 1


StackADT<Integer> listS = new LStack<Integer>(5);
listS.push(10);
listS.push(20);
listS.push(30);
listS.push(40);
listS.pop();
System.out.println(listS);
// Prints: 30 20 10
```

2. Use a stack to implement a parenthesis parser.

   The method **isbalanced** takes a string that can consist of a combination of 3 types of parenthesis: {}, [], (). The method returns True if the expression is Balanced (and False otherwise)

   A balanced expression is an expression where all opening brackets are matched with a closing bracket of the same type, in the correct order.

Some balanced strings are:

1. ( )
2. [( )]
3. { [( ) [ ]] }
4. ({ } [{ }] ( ))
5. The empty string

etc.

Some not balanced strings:

1. (
2.) (
3. (}
4. { [ ]
5. [( ]
6. { } [( ])

etc.

You can assume that all output strings has only the characters { , } , [,] , (,) and spaces (spaces should be ignored)

3. Use a stack to convert a number to binary.
   The method **inBinary** takes a string that represents a decimal number. It should return its binary representation (as a string)

   A decimal number x can be written in decimal using the following algorithm:
   1. Divide the number x by 2 , keep the quotient and the reminder.
   2. Make the quotient the new x, and repeat until the quotient is 0.
   3. Write the remainders in the reverse order, this is the binary number

Example: Write 14 in binary:

14 = 7 * 2 + 0

7 = 3 * 2 + 1

3 = 1 * 2 + 1

1 = 0 * 2 + 1

So 14 in decimal is 1110 in binary, $14_{10} = 1110_2$

Example: Write 67 in binary:

67 = 33 * 2 + 1

33 = 16 * 2 + 1

16 = 8 * 2 + 0

8 = 4 * 2 + 0

4 = 2 * 2 + 0

2 = 1 * 2 + 0

1 = 0 * 2 + 1

So $67_{10} = 1000011_2$

Hint: the static method valueOf(int) in the class String turns a decimal number into a string of its digits.

Part 2: Queues

1. Override toString() for both implementations of QueueADT, where the front value will be printed on the left. (Notice that AQueue uses a circular array)
   toString() do not change the internal state of the queue.

```
QueueADT<Integer> listQ = new LQueue<Integer>(4);
listQ.enqueue(1);
listQ.enqueue(2);
listQ.enqueue(3);
listQ.enqueue(4);
listQ.dequeue();
listQ.dequeue();
listQ.enqueue(5);
System.out.println(listQ);
// Prints: 3 4 5

QueueADT<Integer> arrQ = new AQueue<Integer>(4);
arrQ.enqueue(10);
arrQ.enqueue(20);
arrQ.enqueue(30);
arrQ.enqueue(40);
arrQ.dequeue();
arrQ.dequeue();
arrQ.enqueue(50);
arrQ.enqueue(60);
System.out.println(arrQ);
// Prints: 30 40 50 60
```

2. The method differenceQueue takes a queue of numbers and returns a queue of the differences.
   Example:
   q is the queue (front to back): 7, 10, 3, -5, 4
   Return the queue: -3, 7, 8, -9
   When the method returns q is in it's original state

3. The method mergeQueues takes two sorted queues, and return a queue with all values, sorted.
   Example:
   q1 = 2, 3, 7, 10, 20, 21, 25
   q2 = 1, 5, 8, 9, 14
   Return: 1, 2, 3, 5, 7, 8, 9, 10, 14, 20, 21, 25
   When the method returns q1, q2 are in their original state.