# ASSIGNMENT 1: MATHEMATICAL EXPRESSIONS & GROWABLE DEQUES

**Goal:** The goal of this assignment is to get some practice with stack and queue implementation. One of the goals is also to explore the use of these data structures in the context of an application. In Part (a) of the assignment you have to evaluate mathematical expressions. In Part (b) of the assignment you have to implement the growable array version of the deque data structure.

## ASSIGNMENT 1(A): MATHEMATICAL EXPRESSIONS

**Problem Statement:** You need to evaluate arithmetic expressions over integers that use +, -, *, /, and ^ operators. Here 2^5 denotes 32, the $5^{th}$ power of 2. The expressions can also have parentheses, ( and ), as naturally used in mathematical expressions. All terms of the expression will be space separated. And your goal is to parse them and return the final answer of the expression.

Similar to BODMAS order that you may have studied in the early math classes, our precedence order will be BPDMAS, where P denotes power (^) operation. Here are a few examples to make sure you understand the task

Input: ( 5 + 4 * 3 / 6 – 6 ^ 2 ) ^ 2

Output: 841

Input: 1 / 2

Output: 0.5

Input: ( ( 4 – 4 / 2 ) – 3 ^ ( 3 – 1 ) * 2 ) * 2

Output: -32

In order to implement the expression evaluator you will likely need one or more of linear data structures like stacks and queues that we have learnt in the class. Hint: you can make an evaluator with a constant number of linear data structures.

**Code:** Your code must compile and run on GCL machines. Your code will be run with a parameter inputfile. In this inputfile, there will be a line per expression, and your code should output one line per expression in the same order. If an expression is mathematically invalid it should say “Parse error” or else it should output a single number that is the evaluation of the expressions. For example, if the input is file.txt as follows

( 5 + 4 * 3 / 6 – 6 ^ 2 ) ^ 2

( ( 4 – 4 / 2 ) – 3 ^ ( 3 – 1 ) * 2 ) * 2

5 / 3

( ( 2

And your code is called expression then running “java expression file.txt > output.txt” should create output.txt:

841

-32

1.667

Parse error

## What is being provided?

Your assignment packet contains a formatChecker.jar, sampleInputFile.txt, correctOutputFile.txt and incorrectOutputFile.txt. Running formatChecker.jar with arguments inputfile.txt and output.txt double checks whether your code satisfies the input-output requirements of the assignment. If output.txt contains correct value +-0.001 for all expressions in file.txt, it won't print anything, else for each incorrect value, it will print the line number, corresponding expression in file.txt, expected output and value in the output.txt file. Usage -

java -jar formatChecker.jar sampleInputFile.txt incorrectOutputFile.txt

java -jar formatChecker.jar sampleInputFile.txt correctOutputFile.txt

# ASSIGNMENT 1(B): GROWABLE DEQUEUES

**Problem Statement:** Your second task is to implement a generic ArrayDequeue class. As is to be expected, ArrayDequeue must implement the Dequeue ADT we provided in class. The exact interface will be as follows:

```
public interface DequeInterface {

   public void insertFirst(Object o);

   public void insertLast(Object o);

   public Object removeFirst() throws EmptyDequeException;

   public Object removeLast() throws EmptyDequeException;

   public Object first() throws EmptyDequeException;

   public Object last() throws EmptyDequeException;

   public int size();

   public boolean isEmpty();

   public String toString();

}
```

Your implementation must be done in terms of an array that grows by doubling as needed. Your initial array must have a length of one slot only!

Your implementation must support all Dequeue operations except insertion in (worst-case) constant time; insertion can take longer every now and then (when you need to grow the array), but overall all insertion operations must be constant amortized time as discussed in lecture.

You should provide a toString method in addition to the methods required by the Dequeue interface. A new Dequeue into which 1, 2, and 3 were inserted using insertLast() should return "[1, 2, 3]" while an empty Dequeue should print as "[]".

Ensure that the version of your code you hand in does not produce any extraneous debugging output anymore!

## What is being provided?

Your assignment packet contains two java files – DequeInterface.java and ArrayDequeue.java. You have to implement all the functions in ArrayDequeue class. Main function uses all the functions that you will implement. Remember to run this code (without changes) and make sure that the run is successful.

## What to submit?

1. Submit your code in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" in addition to your code "writeup.txt" should be produced in the working directory.

   You will be penalized for any submissions that do not conform to this requirement.

2. The writeup.txt should have a line that lists names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone say None.
   After this line, you are welcome to write something about your code, though this is not necessary.

## Evaluation Criteria

The assignment is worth 5 points. Your code will be autograded at the demo time against a series of tests. Three points will be provided for correct output and two points will be provided for your explanations of your code and your answering demo questions appropriately.

## What is allowed? What is not?

1. This is an individual assignment.
2. Your code must be your own. You are not to take guidance from any general purpose code or problem specific code meant to solve these or related problems.
3. You should develop your algorithm using your own efforts. You should not Google search for direct solutions to this assignment. However, you are welcome to Google search for generic Java-related syntax.
4. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class.** Please read academic integrity guidelines on the course home page and follow them carefully.
5. Your submitted code will be automatically evaluated against another set of benchmark problems. You get significant penalty if your output is not automatically parsable and does not follow input-guidelines.
6. We will run plagiarism detection software. Anyone found guilty will be awarded a suitable penalty as per IIT rules.