**Task 1:**

Students will implement Stack using linked list. They will code in the file named **ListStack.java**. This java file implements (interface) the **Stack.java** file. Hence they will have to override all the methods. Stack is expected to be covered well in theory therefore no pre lecture on stack is necessary.

**Task 2:**

Getting familiar with java's built in stack class. <https://www.geeksforgeeks.org/stack-class-in-java/>

Students will go over the contents given in the link or any link on java stack. Learn how to create an object of the stack class and use push, pop.

**Task 3:**

Using the built in stack class, students will check **parenthesis order** from an expression. The program will take an expression (could me mathematical or any) with parenthesis from user. Determine the correctness of the expression based on the order of parenthesis. The idea is we will **push** for an opening bracket and **pop** for a closing bracket. After popping check if the popped bracket (opening bracket) is correct for the closing bracket. That is if the popped bracket is '**(**' then the closing bracket in the expression must be '**)**' . Sometimes an expression might have extra opening brackets. If this is the case the stack will never be empty. Sometimes expression might have extra closing bracket. In that case the program will try to pop something from an empty stack. To summarize in one sentence, for en expression to be correct all opening brackets must have their corresponding closing brackets plus after iterating over every character of the expression, the stack must be empty. **Ignore** all other characters. The detailed algorithm is given below.

**Example**:

Expression = 1+b\*[(2/4)-{6+2}]+2, the program should print correct.

Expression = 1+b\*(2/4)-**{**6+2**]**+2, the program should print incorrect.

For an opening curly brace there is a closing square brace.

Expression = 1+b\***[**(2/3)+2, the program should print incorrect.

There is an extra opening brace

Expression = 1+b\*2/4-6**)**+2+2, the program should print incorrect.

There is an extra closing bracket.

**Algorithm**

void parenthesisCheck( ) {

String E = expression from user

for ( i = 0 -- length of E){

if (E[i] is an opening bracket){

push(E[i]);

}

if (E[i] is a closing bracket){

if (stack is not empty){

u = pop()

if (u & E[i] are **not** compatible ){

Opening and closing parenthesis don't match Wrong exp;

break;

}

}else{

Trying to pop from an empty stack Wrong exp;

break;

}

}

if (E[i] is any other character){

ignore;

}

i++;

}

if (stack is empty){

correct expression;

}else{

There is an additional opening bracket Wrong expression;

}

}