**Part A) 10%**
Implement a program that reads in a Java source code file and checks to see if it has balanced {}brackets. Your program should use a stack, implemented as a linked list, to check the brackets as covered in lecture 4. NOTE: you can use a reference called *top* which points to the head of the list.

Your program should run as a command line program and should take a filename as an argument and print one of BALANCED or NOT BALANCED.

For example:

```
c:> java checkBalanced “myProgram.java”
BALANCED
```

**NOTE:** your program should capture and throw appropriate exceptions.

**Part B) 10%**
Write a program that simulates a token ring network. Your implementation should use a circular list where users can be added (Logged On) and removed (Logged Off) from the network. Your program should be broken into two main sections. The first section should add a number of users to the Network. The second section should enter a loop that traverses the circular list and only ends when all users have logged off the network. As the list is traversed each user is asked if they want to log off. The answer to this question should be a random YES or NO. Eventually all user should be logged off the network and the loop should end.

Your system should output to screen each time a user logs on and should also output when a user logs off.

For example:

```
USER: Stephen Sheridan Logged On
USER: Anthony Keane Logged On
:
USER: Anthony Keane Logged Off
USER: Stephen Sheridan Logged Off
[All users logged off.]
```

**NOTE:** your program should capture and throw appropriate exceptions.

**NB:** Check MOODLE for list of users to add to the network.

**Deliverables**
All code should be uploaded to MOODLE and should be demonstrated to me in our lab sessions.

**Submission Date:** Friday 31st of October at 18:00hrs

**All work should be your own. If any evidence of plagiarism is found, all parties involved will receive a grade of zero.**