HUMBER
School of Media Studies
& Information Technology

# GAME255
# DATA STRUCTURES & DESIGN PATTERNS

## LAB 2: JOSEPHUS POSITION WITH DOUBLE CIRCULAR LINKED LISTS

| | |
|---|---|
| **PROFESSOR:** | Jean – Paul Amore |
| **GRADE VALUE:** | 25 Marks = 6% of Final Grade |
| **END DATE:** | Week 4 |

**REQUIREMENTS:**

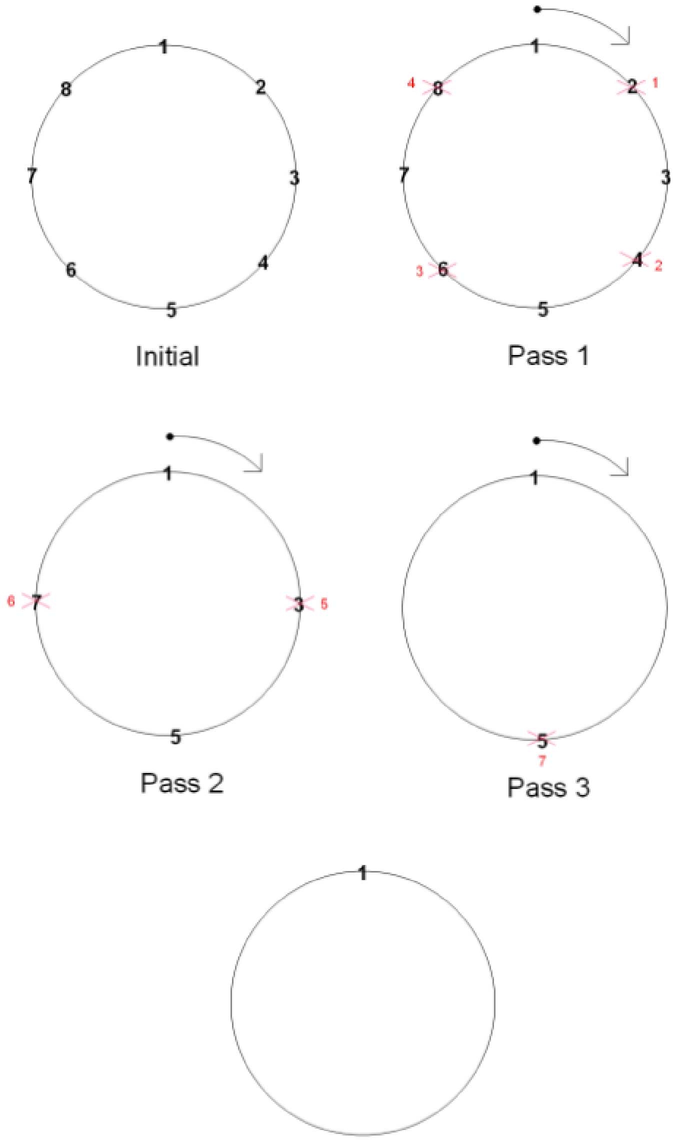
Using the provided project solution, complete the implementation for the member function
`void CDList<T>::getJosephusPosition(int)`.

The Josephus Position assumes there are n people standing in a circle waiting to be executed.

As the execution begins at some point in the circle, it proceeds around the circle in a fixed direction. During each execution, a specified number of people are skipped, and the next person is executed.

The elimination proceeds around the circle as people are executed, and ends when there is one person remaining, who is given freedom.

Using the double circular linked list, determine the position which will grant freedom.



Initial

Pass 1

Pass 2

Pass 3

Final

**DELIVERABLES:**

Submit the Main.cpp file with the implemented method `void CDList<T>::getJosephusPosition(int)`.

| LAB #2 | | | | | | |
|---|---|---|---|---|---|---|
| **CRITERIA** | **0 POINTS** | **1 POINT** | **2 POINTS** | **3 POINTS** | **4 POINTS** | **5 POINTS** |
| **1. LOGIC** | Did not submit | Does not demonstrate ability to use logical process | Poorly demonstrates ability to use logical process | Somewhat demonstrates ability to use logical process | Demonstrates ability to use logical process | Demonstrates exceptional ability to use logical process |
| **2. EFFICIENCY** | Did not submit | Does not demonstrate any efficiency | Poorly demonstrates efficiency | Demonstrates some efficiency | Demonstrates efficiency | Demonstrates exceptional efficiency |
| **3. FUNCTIONALITY** | Did not submit | Asset is not functional | Asset is barely functional | Asset is somewhat functional | Asset is functional | Asset is exceptionally functional |
| **4. PROCESS & ORGANIZATION** | Did not complete assigned work | No code follows a systematic and organized approach to problem solving | Parts of code follows a systematic and organized approach to problem solving | Some code follows a systematic and organized approach to problem solving | Most code follows a systematic and organized approach to problem solving | All code follows a systematic and organized approach to problem solving |
| **5. TEST CASE** | Did not complete assigned work | Does not compile | Barely any code functions with test case | Partially functions with test case | Mostly functions with test case | Fully functions with test case |