Final Examination

Important Notes – PLEASE READ FIRST

- Each submission must be a student's individual work. Collaborations between students are **not** allowed. We use MOSS and other software to detect plagiarism.
- If you adapt code or ideas from a book or website, acknowledge the book (title, edition/year, author(s), page numbers) or website (URL) in your programs or answers. You should not copy the code or text directly and then add some minor modifications, but to borrow ideas only.
- For Questions 4, 5 and 6, you can type the answers using a word processor or text editor and submit the document. You can also write the answers by hand and then scan the pages to submit.
- **Submission instruction:**
  - ○ **websubmit: https://webapp.eecs.yorku.ca/submit/**
  - ○ **submit command: submit 2011Z exam filename**
- Submit each answer/program after you finish it. Submit your work periodically and frequently before the deadline to avoid last-minute technical glitches.
- Do not modify the given class and method definitions. Do not add packages to the submitted Java files. Do not add I/O statements (e.g., scanner, print) to the submitted Java files. *Note*: These mistakes will mess up our automatic grading programs and produce incorrect outputs.

The exam consists of 6 questions which have the same weight.

Question 1

Write a **recursive** Java method that rearranges an array of integers (duplicates allowed) so that all the negative numbers appear before all the non-negative numbers. Note that we do not want to sort the whole array, but only that all the negative numbers appear before all the non-negative numbers.

- Complete method rearrangeArray() in file Rearrange.java. Submit only file Rearrange.java.
- You can write your own main method to test your code, or use the main method in file RearrangeMain.java and input data manually from the standard input for testing.

Hint: Write an iterative algorithm first to help design the recursive version.

Note: Do not sort the array, which takes O(NlogN) given random inputs. Your method should run in O(N) time, where N is the number of elements in the array.

Sample I/O: Given the following array [10, 0, -2, 34, 45, -15, -23, 100, 70, -1, -2], one possible output is [-2, -1, -2, -23, -15, 45, 34, 100, 70, 0, 10]. Note that your output may be different depending on your algorithm, but all the negative numbers must appear before all the non-negative numbers in the final array.

Question 2

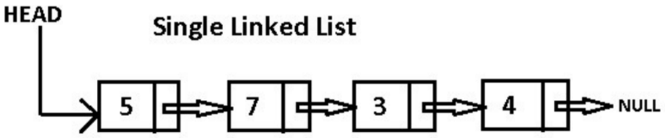
Assume a singly linked list storing integers (duplicates allowed) and having a “head” pointer pointing to the first element of the list (see diagram below). Write a **recursive** method named numOfTimes() that returns the number of times an integer k appears in the linked list. The method returns 0 if the linked list does not contain integer k or is empty.

- Complete method numOfTimes() in file LinkedList.java. Submit only file LinkedList.java.
- You can write your own main method to test your code, or use the main method in file LinkedListMain.java and input data manually from the standard input for testing.

Hint: Write an iterative algorithm first to help design the recursive version.

Note: Your method should run in O(N) time, where N is the number of elements in the list.

Sample I/O: Given the following list (10, -50, 20, 8, -12, 20, 45, 20, -39, 68) and integer k = 20, method numOfTimes() returns 3. That is, number 20 appears 3 times on the list.



Question 3

Given a binary tree that stores distinct integers (no duplicates), write a Java method that returns true if the tree contains number zero (0), and false otherwise. The method returns false if the tree is empty.

- Complete method hasZero() in file BinaryTree.java. Submit only file BinaryTree.java.
- You can write your own main method to test your code, or use the main method in file BinaryTreeMain.java and input data manually from the standard input for testing.

Note: These are general binary trees, not binary search trees. In the implementation of binary trees in file BinaryTree.java, the external nodes are not dummy nodes but contain actual data (integers).

Sample I/O: Given a tree contains the following integers (30, 50, -12, 0, 142, 8, -67, -109, 25), method hasZero() returns true.

Question 4

This question refers to Problem 1 "Circular Arrays" of Assignment 3. Assume a method named buildQ() that is used to build a queue of size N. Each element is added to the queue one by one using method insertLast(). Every time the array is full, buildQ() extends the array by doubling its capacity and inserts the element into the new array as specified in the assignment. Obtain a **θ**

running time bound for method buildQ() as a function of N, assuming the most efficient implementation and N = $2^{\mathbf{k}}$ where k is a large positive integer.   Show detailed calculations.
Note: Name the submitted file q4.pdf, q4.doc, q4.docx, q4.txt, or q4.jpg, etc.

Question 5

This question refers to Assignment 4, “AVL Trees”. Assume a method named buildAVL() that is used to build an AVL tree by inserting N keys (entries) into the tree one by one using method insert(). Note that the tree must be rebalanced after each insertion. Obtain a **θ** running time bound for buildAVL() as a function of N. Show detailed calculations.
Note: Name the submitted file q5.pdf, q5.doc, q5.docx, q5.txt, or q5.jpg, etc.

Question 6

Solve the following recurrence by obtaining a **θ** bound for **T(N)** given that **T(1) = θ(1)**. Show detailed calculations.

T(N) = T(N/2) + N – 1

Note: Name the submitted file q6.pdf, q6.doc, q6.docx, q6.txt, or q6.jpg, etc.