Name: ______________________ Student Number: ________________

Before you start answering questions:

- Read these instructions.
- Put your name and student number at the top of this page.
- ***Turn off your phone.***
- Put your name and student number on the front of an exam booklet.

-

The Rules:

- Do not start or open this exam until you are instructed to do so.
- This is an open book exam. You can use your textbook as well as any papers or notes that you feel will be helpful.
- You may not use any electronic devices except a basic non-programmable calculator.
- You may not share any notes or papers or books with neighbors or friends.
- No computers or phones or MP3 players are allowed.
- No talking during the exam.
- This is a 2 hour exam, but you may take 3 hours if you wish (I don't want time to be a factor).
- You must stay for at least 1 hour of the exam.
- Record your answers in a separate exam booklet.

When you are finished:

- Give your exam ***and*** your exam booklet to the instructor.
- Quietly leave the room.

Instructions

Read the questions carefully. Ask me to clarify the question if you don't understand it or if something seems strange. It is possible that there is a mistake in the wording of the question.

This exam contains 7 questions. Each question is worth 10 marks. The maximum score on the exam is 70/70.

As with all exams, it is your responsibility to convince the marker that you know what you are doing. If your work is not understandable or is too messy for the marker to read, it will be assigned zero (0) marks.

1. Draw the 11-element hash table resulting from hashing the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, using the hash function h(*i*) = (2*i*+5) mod 11. Assume collisions are handled by chaining.

2. Assume that you have a hash table that has *m* buckets. Collisions are handled by chaining. The table currently contains *n* elements, where $n \neq m$. Provide the pseudo-code for an algorithm to convert the hash-table to a sorted array of size *n*. Calculate the worst case efficiency of your algorithm. Describe the type of input that will result in worst case performance for your algorithm.

3. Assume that you have a sorted array containing the elements 2, 3, 11, 9, 7, 12, 29, 13, 33, 34, 35, 22, 23, 44, 41, 61, 88, 89. Write out the pseudo-code for a non-recursive binary search, and then show the values of the variables in your algorithm after each iteration if the search key is K=25. Would your algorithm have terminated sooner if the number 25 was in the array? Provide a convincing argument for your answer.

4. Assume that you have an unsorted doubly linked list with header and trailer sentinels. Write the pseudo-code for an *O(n/2)* link-hopping algorithm to insert a new item into the middle of the list. Your solution cannot make use of counters or the List.size() method.

5. Assume that you are given a sorted array of integers A[*0..n-1*]. Describe a *O(log n)* divide and conquer algorithm to determine whether there exists an index *i* such that A[*i*] = *i*. For example, there is no such *i* in A=[-3, 5, 9, 10], but there is such an *i* in A=[-1, 0, 2, 6] (A[2]=2). Provide pseudo-code for your algorithm.

Continued next page …

6. Consider the algorithm shown below. This is a standard algorithm to merge two sorted lists, maintaining the sorted order.

   a. Identify the basic operation in the algorithm.

   b. How many times is the basic operation executed?

   c. What is the efficiency class of this algorithm? (justify your answer)

   d. Suggest an improvement to the algorithm, and indicate your improved algorithms efficiency class.

```
// input two sorted arrays a[0..n-1], b[0..m-1]
// the merged lists are returned as a new list c[0..m+n-1]
//
Merge(a, b)

n = a.size
m = b.size

if n = 0   return b
if m = 0   return a

while (i < n) and (j < m)
   if a[i] ≤ b[j]
      AddToResult(c, a[i])
      i ← i + 1
   else
      AddToResult(c, b[j])
      j ← j + 1
while i < n
   AddToResult(c, a[i])
   i ← i + 1
while j < m
   AddToResult(c, b[j])
   j ← j + 1

return c

// find the posn of the item to be added, and add the item
// note result.size returns the number of elements in result
//
AddToResult(result, item)

i ← 0
while (result(i) ≤ item) and (i < result.size)
   i ← i + 1
result(i) ← item
```

7. Consider the java program shown below.

   a. Explain what the program does.

   b. Identify the basic operation in the algorithm.

   c. How many times is the basic operation executed?

   d. What is the efficiency class of this algorithm? (justify your answer)

```
public static void mystery(List list, Random rnd) {
    for (int i = list.size(); i > 1; i--)
        idiom(list, i - 1, rnd.nextInt(i));
}

public static void idiom(List a, int i, int j) {
    Object tmp = a.get(i);
    a.set(i, a.get(j));
    a.set(j, tmp);
}
```

And from the List class …

```
public Object get(int index) {
    ListNode currentNode = head;
    for (int i = 0; i < index; i++)
        currentNode = currentNode.next;
    return currentNode.contents();
}

public void set(int index, Object item) {
    ListNode currentNode = head;
    for (int i = 0; i < index; i++)
        currentNode = currentNode.next;
    currentNode.set(item);
}
```