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.
- Place all your belongings on the floor, except your pens/pencils, notes (1 page), erasers, rulers.

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 7-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*) = *i* mod 7. 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 m ≈ n. Provide the pseudo-code for an algorithm to resize the hash-table to twice its original size, and move all *n* elements to the new hash table. Calculate the worst case efficiency of your algorithm. Describe a scenario that would result in worst case performance for your algorithm.

3. Assume that you have an array *A* = [7,3,9,4,2,5,6,1,8]. Write out the pseudo-code for a bubble sort, and then show the results of the sort after each pass. Would your algorithm have used fewer basic operations if the input was ordered 1,2,3,9,4,5,6,7,8? Provide a convincing argument for your answer.

4. Assume that you have a doubly-linked circular list (ie: the *next* pointer of the last node points to the first node). The list contains *n* elements, where *n* is an even number. Write the pseudo-code for an *O(n/2)* algorithm that splits the list into two smaller, equally sized, circular lists.

5. Describe a divide and conquer algorithm to search a 2D array of size 2^i x 2^i for a search key *K*. You can assume the array stores integers. Provide pseudo-code for your algorithm.

Continued next page …

6. Consider the algorithm shown below.
   a. Explain what the algorithm 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?

```
// P is an array of char P[0..m]
// T is an array of char T[0..n]
//
Main(P, T)
   n ← T.size
   m ← P.size
   r ← Locate(P, T, 0, n)
   while r ≠ -1 {
      s ← Flip(P, T, r)
      r ← Locate(P, T, s, n)
   }

Locate(P, T, s, n)
   if s ≥ n
      return -1
   for i ← s to n-m {
      j ← 0
      while j<m and P[j] = T[i+j] {
         j ← j+1
      }
      if j = m return i
   }
   return -1

Flip(P, T, r)
   x ← P.size
   while x ≥ 0 {
      T[r] ← P[x]
      x ← x-1
      r ← r+1
   }
   return r
```

Continued next page …

7. Consider the java methods shown below, which implement the `contains(String)` method of a BucketSort algorithm.

   a. Identify the basic operation in this program.

   b. How many times is the basic operation executed?

   c. What is the efficiency class of this program?

   d. Suggest an improvement to the algorithm, and indicate your improved algorithms efficiency class. A textual description of your improvement will suffice (ie: you do not need to write pseudo-code or java).

```
public boolean contains( String item )
{
   int bucket = hash(item);

   if ( hashTable[bucket].indexOf(item) == -1 )
      return false;
   else
      return true;
}

public int indexOf(Object item)
{
   ListNode currentNode = head;
   for (int i = 0; currentNode != null; i++)
   {
      if ( currentNode.contents().equals(item) )
         return i;
      else
         currentNode = currentNode.next;
   }
   return -1;
}

protected int hash(String word)
{
   int bucket = (int) (((computeK((String)word)-1)*numBuckets) / maxK);
   return bucket;
}

private double computeK(String word)
{
   double value = 0;
   int cValue = 0;
   int power = maxLength-1;
   int maxChar = Character.getNumericValue('Z');

   for (int i=0; i<word.length(); i++) {
      if (Character.isLetterOrDigit(word.charAt(i))) {
         cValue = Character.getNumericValue(word.charAt(i))
         value = value + (cValue * Math.pow(maxChar,power));
      }
      power--;
   }
   return value;
}
```