Name: _____ **Student Number:** _____

Before you start answering questions:

- Put your name and student number at the top of this page.
- ***Turn off your phone.***
- Put all belongings on the floor.

The Rules:

- Do not start or open this exam until you are instructed to do so.
- This is a closed book exam.
- You may use a single 8.5"x11" hand-written hint sheet.
- ***Answer the questions directly in this copy of the exam.***
- Do not share notes or papers with neighbors or friends.
- You may not use any electronic devices.
- No calculators, computers, phones, or MP3 players.
- No talking during the exam.
- This is a 120 minute (2hr) exam.
- You must stay for at least 1 hour of the exam.

When you are finished:

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

Question	Grade
1	/ 10
2	/ 10
3	/ 10
4	/ 10
5	/ 10
6	/ 10
Total	/ 60

Instructions

Read the questions carefully. Ask the instructor to clarify a 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.

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. A hash table has 5 buckets and uses the hashing function **h(*i*) = (4*i*+3) mod 5**. Collisions are handled by chaining. Draw the hash table resulting from hashing the following keys:

10, 3, 6, 11, 5, 1.

2. Consider the array **A=[8, 21, 18, 25, 9, 13, 28]**.

   a. Use either the Heapify or the HeapBottomUp algorithm to construct a max heap H[] from array A[]. Show your work, as well as the final H[] (as it is stored in an array).

   b. Sort the data into ascending order by applying heapsort to your heap H[]. Show the array H[] after each iteration of the heapsort.

3. Consider the following algorithm, which is pseudocode for the **`String.replace()`** method from the Java 6 SDK.

   a. Identify the basic operation in the algorithm.

   b. In the worst case, how many times is the basic operation executed?

   c. What is the worst-case efficiency class of this algorithm?

```
1  algorithm: replace(oldChar, newChar)
2      if (oldChar ≠ newChar)
3          i ← -1
4          char[] val ← theString          // treat the string as a
5          len ← theString.length()        // ... character array

6          while (++i < len)
7              if (val[i] = oldChar)
8                  break

9          if (i < len)

10             allocate space for buf[len]  // a temp buffer
11             for (j ← 0 ; j < i ; j++)
12                 buf[j] ← val[j]

13             while (i < len)
14                 c ← val[i]
15                 if (c = oldChar)
16                     buf[i] ← newChar
17                 else
18                     buf[i] ← c
19                 i++
20             end while

21             return buf[]
22         end if

23     end if
24     return theString
```

4. An auto parts store verifies and updates their inventory every year. Luckily everything in the store is barcoded. All they have to do is walk around with a barcode scanner and scan each item that is on a shelf or in the warehouse. You have been asked to write a program that will report the quantity of each item that is in stock.

   Input to your program is from two files. The first file contains a list of the *n* inventory items the store sells, along with the corresponding barcode. The second file contains a list of *m* barcodes that were scanned.

   Output is a new file containing one line for each inventory item, along with the quantity currently in stock. The output is sorted by decreasing quantity on hand.

   Sample Input

   File 1: part_descriptions.txt
   ```
   Bumper 101224
   Headlight 223433
   Manifold 123456
   Solenoid 656565
   Tire 001998
   ```

   File 2: scanned_codes.txt
   ```
   101224
   223433
   001998
   223433
   223433
   123456
   101224
   ```

   Sample Output

   File 3: quantity on hand.txt
   ```
   Headlight 3
   Bumper 2
   Manifold 1
   Tire 1
   Solenoid 0
   ```

   a. What are the major data structures that you will use to solve this problem? Justify your choices, and explain what they will store.

   b. Outline (in English or very high level pseudocode - no java syntax allowed!) an ***efficient*** algorithm to solve this problem using the data structures you identified.

   c. What do you expect the worst case efficiency of your solution to be? Justify your answer.

5. Let A = $\{a_1,..., a_n\}$ and B = $\{b_1, ..., b_m\}$ be two sets of numbers. Consider the problem of finding their intersection, i.e., the set C of all the numbers that are in both A and B.

   Provide detailed pseudocode for an algorithm to solve this problem in ***better*** than $O(n^2)$ time.

6. 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 detailed pseudo-code for your algorithm.

Spare page.