CISC/CMPE-365*
Test #2
October 21, 2016

Student Number (Required) ______________________

Name (Optional)________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be reconsidered after the test papers have been returned.

The test will be marked out of 50.

Question 1	/12
Question 2	/25
Question 3	/12
Question 4	/1
TOTAL	/50

General marking philosophy: a student who gives enough of an answer to show they understood what they were supposed to do, even if they couldn't do it (or made lots of errors while doing it) should get at least 50% on that question.

Full marks should be given if a solution is sound and not missing anything important.

Feel free to give marks like 9.5/10 to a solution that is correct but contains a minor error.

Students may come up with solutions that are completely different from mine but still completely correct. Correct solutions should get full marks even if they don't match mine.

Students should always get a few marks for trying a question. The only way to get a 0 is to leave the page blank or write something completely irrelevant.

Question 1 (12 Marks)

(a) **[6 marks]** Show the bitstring codes that result from applying the Huffman Coding algorithm to a string containing the following set of letters with the indicated frequencies:

a	b	c	d	e	f	g	h
1	1	2	3	5	8	13	21

Solution: most students will probably draw the tree to explain how they get the final bitstrings. The tree will probably look something like this

```
a      b      c      d      e      f      g      h
  \   /      /      /      /      /      /      /
   ab      /      /      /      /      /      /
      \   /      /      /      /      /      /
       abc     /      /      /      /      /
         \   /      /      /      /      /
          abcd    /      /      /      /
            \    /      /      /      /
             abcde    /      /      /
               \    /      /      /
                abcdef    /      /
                   \    /      /
                    abcdefg   /
                        \   /
                        abcdefgh
```

with the edges labeled "0" and "1" . It is not necessary to label the internal vertices. If the "up and left" edges are "0" and the "up and right" edges are "1" this gives

a: 0000000 b: 0000001 c: 000001 d: 00001 e: 0001 f: 001 g: 01 h: 1

(b) **[3 marks]** Is your answer in (a) unique? Why or why not?

No, it is not unique. Exchanging the "0" and "1" labels on any pair of edges that both go "up" from a single vertex with give an equivalent code. It is also true that swapping all "0" and "1" edge-labels will give an equivalent code.

(c) **[3 marks]** Generalize your answer from (a) to describe an optimal prefix-property code when the letter frequencies are the first **n** Fibonacci numbers.

The highest frequency letter gets a bitstring of length 1. The second highest frequency letter gets a bitstring of length 2, and so on down to the last two letters, which both get bitstrings of length n-1. The bitstrings must obey the prefix-property rule.

If a student does not know the definition of the Fibonacci sequence ...if they extrapolated from part (a) in a plausible fashion and gave a decent answer, that's ok

Question 2 (25 marks)

Suppose you have **K** dollars in your pocket, and you want to buy Hallowe'en candies to give to trick-or-treaters. At the candy shop there are **n** small buckets of different types of candy. Each piece of candy is priced at $1, so you can only buy a maximum of K pieces of candy. **For each type of candy, you have a satisfaction value** that you experience from giving one piece of that candy to a trick-or-treater.

(a) **[10 marks]** Suggest a Greedy Algorithm to **maximize the total satisfaction** you will experience when you give away all the candy that you buy.

```
Sort the candies in descending order by their satisfaction value.
$R = K$
while $R > 0$ and there are still some candies left to buy:
    buy a candy with the highest satisfaction value
    $R = R – 1$
```

or

```
Sort the candies in descending order by their satisfaction value.
$R = K$
while $R > 0$ and there are still some candies left to buy:
    buy as many candies as possible with the highest satisfaction value
    $R = R$ – the number of candies bought on the line above this one
```

(b) **[5 marks]** Prove that your algorithm's first choice is optimal (i.e. that there is an optimal solution that makes the same choice)

Consider an optimal solution that does not include as many of the highest satisfaction value candies as the Algorithm's solution. Then we can replace some equal-or-lower value candies in the optimal solution, without lowering its total value, using the left-over highest value candies. Thus there is an optimal solution that matches the number of highest-value candies in the Algorithm's solution.

(c) **[10 marks]** Complete the proof that your algorithm finds an optimal solution to the problem.

Continuing the argument above, we can start with an optimal solution that matches the Algorithm's choice with respect to the highest-value candy. Using the same reasoning, we can find an optimal solution that also matches the Algorithm's choice with respect to the second-highest-value candy, and so on. Eventually we reach an optimal solution that is identical to the Algorithm's solution ... hence the Algorithm's solution is optimal.

Proof by induction is also a reasonable approach.

Marking: students seem to have interpreted this problem in a variety of ways. For example, some students assumed the buckets are sealed and you can't pick individual candies out. This obviously changes the answer, but they can still come up with a greedy algorithm (although it won't always give the optimal solution because this interpretation makes the problem equivalent to the 01 Knapsack Problem). Other students assumed that the small buckets contain infinite numbers of candies (?) … which also affects the details of the answer, but not its principle. If they give an answer that is correct relative to their interpretation, that's ok.

Question 3 (12 Marks)

Is Dijkstra's Algorithm for finding least-weight paths in a graph with positive edge-weights a Greedy Algorithm?   Why or why not?

Case for Yes: on each iteration, the algorithm chooses the best option available to it. It never looks forward to anticipate future choices or back to revisit previous choices. This is the essence of the Greedy strategy.

Case for No: Greedy algorithms are supposed to start with a sort. Dijkstra's Algorithm does not start with a sort … so it is not a greedy algorithm

Marking: I'm willing to accept either "Yes" or "No" for this … but they have to give a decent reason for their answer.

Question 4 (1 mark)

Consider the following Greedy Algorithm for CNF-SAT:

```
sort the boolean variables in the expression in descending order based on
    how many terms they occur in

for each boolean variable, set it to True unless its negation has already
    been set to True
```

True or false: If E is a satisfiable expression in CNF form, this algorithm will always find a truth assignment that satisfies E

true

FALSE

The correct answer is False