CS 150 Test 1 – Spring 2008 - KEY

Name: _Mark Sherriff_______

Email ID: ___mss2x_____

Pledge: (You must pledge the test before we will grade it.)

This test is closed note, closed book. It is a 50-minute test. You are not to speak with anyone except the instructor or a teaching assistant for any reason except an emergency during the test. ***Important:*** Students may take this test at different times. It is a violation of the Honor Code to talk about the test in any way with a student who has not yet taken it.

The questions may not be in any particular order. Hard questions may be mixed with easy questions. If you start a question and it's hard or time-consuming, you might want to save it for the end of the test and go ahead and work on easier problems. State any additional assumptions you make. Good luck!

PROBLEM	MAX	SCORE
1	15	
2	20	
3	20	
4	10	
5	5	
6	5	
7	5	
8	5	
9	5	
10	5	
11	5	
12	5	
TOTAL	100	

Question 1 - BNF [15 points]

Write a BNF grammar that could have generated the following sentences:

```
Flying over the mountain, Superman could see a vast, empty, desolate
plain where the spaceship could have crashed.

Driving over the curvy road, Prof. Layton and his apprentice Luke
could see a small, quaint village where numerous people lived.

Flying over the deep ocean, Prof. Sherriff and his wife could see a
vast ocean where it would be most unfortunate to crash.

Walking over the Lawn, Thomas Jefferson could see a small academical
village where students would live without bathrooms.
```

Remember: you must have a non-terminal to start from. Points will be awarded based upon finding the most flexible grammar (i.e. all the appropriate non-terminals). You may use either the traditional BNF syntax or the syntax from HW1 - just make it clear what your terminals and non-terminals are.

<phrase> ::= <verb> over the <noun>, <person> could see <place> where <situation>.

<verb> ::= Flying | Driving | Walking

<noun> ::= <adjs> mountain | <adjs> road | <adjs> ocean | <adjs> Lawn

<person> ::= Superman <partner> | Prof. Layton <partner> | Prof. Sherriff <partner> | Thomas Jefferson <partner>

<partner> ::= and his apprentice Luke | and his wife | epsilon

<place> ::= <adjs> plain | <adjs> village | <adjs> ocean

<situation> ::= the spaceship could have crashed | numerous people lived | it would be most unfortunate to crash | students would live without bathrooms.

<adjs> ::= <adj>, <adjs> | <adj> | epsilon

<adj> ::= curvy | deep | vast | empty | desolate | small | quaint

Rubric:
+10 points - working grammar
+5 points - degree of flexibility (+3 points for nested non-terminals, +2 for recursively repeating non-terminals)

Question 2 – Algorithmic Thinking [20 points]

Write a **recursive** algorithm that will solve the following problem. Please put it in a step-by-step form, either numbering steps or using pseudo-code - or a programming language if that works better for you.

Ignoring spaces, punctuation, and capitalization, given any phrase can you tell me if it is a palindrome or not? (A palindrome is a phrase that reads the same backwards and forwards, like "race car.")

Make sure to be very clear about every step in the process!

2 basic algorithms:

1) **use the reverse function from the homework**
2) **compare the first and last letters - if the same, recursively call on the remaining string**

Rubric:
+10 points - working algorithm
+10 points - recursive algorithm

Question 3 – Complexity [20 points]

a) Identify the critical section of your algorithm above.

Either the comparison or the concatenation step

+6 if correct
+3 if very close

b) What is the “Big Oh” complexity of your algorithm? Why?

O(n) - because you only have to pass through the string once

+6 if correct
+3 if very close

c) Draw a picture illustrating the complexity growth rates of f(n), g(n), h(n), and q(n):
f(n) is in O(g(n))
f(n) is in Θ(h(n))
f(n) is in Ω(q(n))

Check the notes / slides for the answer here.

Basically, g(n) over f(n) (which is the same line as h(n)), over q(n)

+2 for each line

Question 4 – Sorting   [10 Points]

What is the “Big Oh” complexity of mergesort? ______**n log n**____________________

Perform a mergesort on the following array. Show all steps. **Underline the base case.**

```
[5, 3, 9, 10, 29, 41, 1, 3]
```

5, 3, 9, 10   29, 41, 1, 3

5, 3   9, 10   29, 41   1, 3

5   3   9   10   29   41   1   3

3,5   9,10   29,41   1,3

3, 5, 9 ,10   1, 3, 29, 41

1, 3, 3, 5, 9, 10, 29, 41

Rubric:
+2 for complexity
+2 for underline
+4 for correct steps

Short Questions – We will drop the lowest score of these questions

Question 5 – Complexity [5 points]

Explain why the Fibonacci algorithm takes much longer recursively than iteratively.

recursive has no memory - has to repeated calculate each step - iterative is a "sliding window"
+5 good answer
+3 okay answer
+1 something that resembles an answer

Question 6 – Fractals [5 points]

Describe two ways in which fractals can be recursive.

1) drawn or 2) recursive image

+3 for one, +5 for both

Question 7 – Haskell [5 points]

Identify the following parts of the Haskell function: a) parameters, b) return type, c) function name, d) base case, and e) recursive call.

```
power :: Int -> Int -> Int
power x 0 = 1
power x n = x * power x (n-1)
```

a) either the first two Ints or the x 0 / x n on the left, b) the last Int, c) power on the left, d) second line, e) power x (n-1)

+1 point each

Question 8 – History [5 points]

How did Charles Babbage and Ada Lovelace know each other? What did they do that was so important?

Two British mathematicians that collaborated. Babbage created the differencing engine, Ada created the first programming language for it.

+5 good answer
+3 okay answer
+1 something that resembles an answer

Question 9 – Design [5 points]

Define and give an example of a "strange loop."

Loop that as it's going does not appear that it should come back but it does against all expectations - examples being Escher's waterfall, staircase, hands or Bach's canons, etc.

+3 define, +2 example

Question 10 – Searching [5 points]

Compare and contrast iterative search with binary search.

iterative - O(n) - straight down the line, compares with everything
binary - O(lg n) - splits up the list in half each time - must be sorted

+5 good answer
+3 okay answer
+1 something that resembles an answer

Question 11 – Haskell [5 points]

What would happen if I ran `totalsum 20` here? Why? What about `totalsum 30`? Why?

```
totalsum :: Int -> Int
totalsum 1 = 1
totalsum 30 = 400
totalsum n = n + totalsum (n)
```

`totalsum 20 - will never end because there is no n-1`

`totalsum 30 - returns 400 because of pattern matching`

+5 good answer
+3 okay answer
+1 something that resembles an answer

Question 12 – Complexity [5 points]

Put these in order from least to most complex.

O(2^n)

O(n^3 + 2n + 400)

O(4^10)

O(n lg n)

O(lg n)

O(n^2)

O(n)

O(4^10), O(lg n), O(n), O(n lg n), O(n^2), O(n^3), O(2^n)

-1 per individual mixup

_________ __________ __________ ____________ __________ ___________ ____________

`<- Least Complex                                        Most Complex ->`

Optional Course Survey

You may separate this portion from the test and submit it anonymously. Your feedback is essential to ensuring that the course staff is addressing any issues in the course.

How was the test?

What things have you liked about the course so far? What would you like to see continue through the rest of the semester?

Any concerns or suggestions?

What topics would you like to see in the next section of the course?