CS 134 Midterm
Fall 2006

This is a closed book exam. You have 50 minutes to complete the exam. There are 5 questions on this examination. The point values for the questions are shown in the table below. Your answers should fit in the space provided in the exam booklet. Paper for scrap work will be made available during the examination.

Good Luck!

NAME: ______SAMPLE SOLUTION_______
LECTURE SECTION: __ 9AM __ 10AM

Question	Points	Score
1	9	
2	9	
3	11	
4	11	
5	10	
TOTAL	50	

I have neither given nor received aid on this examination.

Signature: ___________________________

1) On almost any item you buy these days you will find a marking like the one shown on the right known as a UPC (UPC stands for Universal Product Code).

The black and white bars in a UPC are simply an encoding of the twelve decimal digits that appear below them (051000012517 in this case). The bars encode this information using a system designed to be easily read by devices attached to cash registers at super market checkout counters.

0 51000 01251 7

It should not surprise you to learn that the encoding scheme used is based on binary. In fact it is basically a form of on-off keying. A black bar of a certain width represents a 1 and a white region of the same width represents a zero. Below, we show a narrow slice of the UPC shown above and the sequence of binary digits that would be "read" when a UPC reader at a checkout counter scanned this UPC.

10100011010110001001100100011010001101000110101010111001011001101101100100111011001101000100101

If you were crazy enough to count them, you would notice that there are 95 binary digits encoded in this UPC. For this question, we would like you to consider how this compares to the minimum number of bits needed to encode 12 decimal digits.

a) Suppose we treated the 12 decimal digits (051000012517) as a 12-digit base-10 number. How many bits would be required? That is, how many binary digits would be required to encode any 12-digit decimal number as efficiently as possible?

The following table of powers of two should prove helpful for this part.

$2^{29} = 536{,}870{,}912$	$2^{35} = 34{,}359{,}738{,}386$
$2^{30} = 1{,}073{,}741{,}824$	$2^{36} = 68{,}719{,}476{,}736$
$2^{31} = 2{,}147{,}483{,}684$	$2^{37} = 137{,}438{,}953{,}472$
$2^{32} = 4{,}294{,}967{,}296$	$2^{38} = 274{,}877{,}906{,}944$
$2^{33} = 8{,}589{,}934{,}592$	$2^{39} = 549{,}755{,}581{,}888$
$2^{34} = 17{,}179{,}869{,}184$	$2^{40} = 1{,}099{,}511{,}627{,}776$

12 decimal digits require 40 binary digits since $2^{39} < 10^{12} < 2^{40}$

b) From part (a) you should now know that 12 decimal digits could be encoded using far fewer than 95 bits! Part of the reason so many bits are used in the UPC is that some of the bits shown are always the same regardless of the decimal digits being encoded. In particular, the first three bits are always "101", the last three bits are always "101" and the sequence "01010" always appears in the middle of the UPC.

Given our discussions of techniques for encoding information in binary (on-off keying, manchester encoding, start bits, etc.) and the devices you have seen that scan UPC codes in stores, speculate as to the purpose of these fixed sequences of 1's and 0's.

The fixed strings of alternating 0's and 1's probably provide the ability to synchronize the receiver's "clock" with the rate at which the UPC is being scanned. The bits at the beginning and end may also serve as an extended start bit like the SYNC field of an Ethernet packet.

c) It turns out that in a UPC, each decimal digit is encoded separately, using an independent subsequence of the binary digits (instead of treating all 12 digits as a single number). Once the fixed bits discussed in part (a) are accounted for, there are still 84 binary digits being used. That means there are 7 binary digits for each decimal digit. How does this compare with the most efficient such encoding. That is, what is the smallest number of binary digits that would be sufficient to encode any single decimal digit?

A single digit requires 4 bits since $2^3 < 10 < 2^4$.

2) Consider the following simple Java program (with the import statements removed to save space). It displays a button and a text field on the screen. Initially the button's label is "Change Label". Each time the button is clicked, this label is replaced by whatever text is currently in the text field.

```
public class UnderstandingDecls extends GUIManager {

    private JButton change = new JButton( "Change Label" );
    private JTextField newLabel = new JTextField( 10 );

    public UnderstandingDecls() {
        this.createWindow( 200, 300 );

        contentPane.add( change );
        contentPane.add( newLabel );
    }

    public void buttonClicked( JButton which ) {
        change.setText( newLabel.getText() );
    }

}
```

On the following pages, you will find several programs that are similar but not identical to the program shown above. The differences all involve the placement of declarations and assignment statements. Some of these programs will function exactly like the program shown above. Others will work differently or not at all.

For each program, indicate whether a) it will work like the original program, b) a syntax error will be detected as soon as you compile the program, c) the program will run, but it will not behave like the original (possibly including causing an error to occur while it runs). If there will be an error, identify the line on which it will occur and explain the nature of the error. If no error occurs, but the program behaves differently from the original, explain how it differs.

a)

```
public class UnderstandingDecls2 extends GUIManager {

    public UnderstandingDecls2() {
        this.createWindow( 200, 300 );

        JButton change;
        JTextField newLabel;

        change = new JButton( "Change Label" );
        newLabel = new JTextField( 10 );

        contentPane.add( change );
        contentPane.add( newLabel );
    }

    public void buttonClicked( JButton which ) {

        JButton change;
        JTextField newLabel;


        change.setText( newLabel.getText() );     <------------
    }

}
```

The revised program will not be run because Java will identify a syntax error on the last line of the buttonClicked method. Because change and newLabel are defined as local variables in both the constructor and the buttonClicked method, the assignments to these names in the constructor do not associate meanings with the versions of the name declared in the method. Therefore, Java will recognize that these names will be undefined when they are used in the method.

b)

```
public class UnderstandingDecls3 extends GUIManager {

    private JButton change = new JButton( "Change Label" );
    private JTextField newLabel;

    public UnderstandingDecls3() {
        this.createWindow( 200, 300 );

        contentPane.add( change );

        newLabel = new JTextField( 10 );
        contentPane.add( newLabel );
    }

    public void buttonClicked( ) {
        change.setText( newLabel.getText() );
    }

}
```

This version will work just like the original. The only difference is that this version waits to create the JTextField and assign it to the name newLabel until just before it is first used.

c)

```
public class UnderstandingDecls4 extends GUIManager {

    private JButton change = new JButton( "Change Label" );
    private JTextField newLabel = new JTextField( 10 );

    public UnderstandingDecls4() {
        this.createWindow( 200, 300 );

        contentPane.add( new JButton( "Change Label" ) );
        contentPane.add( new JTextField( 10 ) );
    }

    public void buttonClicked( JButton which ) {
        change.setText( newLabel.getText() );
    }

}
```

This version will run, but it won't work as expected. The button and textfield that will be displayed on the screen will not be the button and textfield associated with the variable names because these names are not used when the GUI components are added to the content pane. Therefore, when someone clicks on the button, the setText operation will change the button associated with the name change, but because this button is not visible on the screen, nothing in the program's window will change.

3) There are several parameters in the design of the Ethernet protocol that are interrelated in important ways. For example, the minimum packet size is chosen so that the time required to transmit a minimum size packet will equal or exceed twice the time required for a signal to travel from one end of the network to the other. As a consequence, if the network designers had wanted to use a smaller minimum packet size, they would also have to change the value specified as the maximum length of a network.

Assume that the following variables are used to describe the parameters of an Ethernet:

R = the rate at which bits are transmitted (measured in bits per second. This is what the Ethernet paper calls C)

L = the maximum allowed length of a network (measured in meters)

c = the speed at which signals propagate on network cables (measured in meters/second --- close to the speed of light)

W = the expected number of slots of contention before a successful transmission

Q = the number of active stations attached to the network

M = the minimum allowable packet size (measured in bits)

a) Give a formula for M in terms of the other variables above.

$M = R \times 2 \times L / c$

(The statement of this problem was actually slightly flawed. Given the problem statement, the equal sign above should actually be a > sign. In practice, however, M is chosen by rounding up the formula given above to the nearest integer, so the equal sign comes close to reflecting reality.)

b) The slot time used in the exponential backoff algorithm must also chosen so that it equals or exceeds twice the time required for a signal to travel from one end of the network to the other. This ensures that two stations will only collide if they randomly choose to delay their transmissions for exactly the same number of slot time. Give a formula for the minimum length of the slot time, T, in terms of M.

$T = M / R$

c) Give a formula for T in terms of the variables listed above without using M.

$T = 2 \times L / c$

d) While the original Ethernet standard used a transmission rate (R) of 10 million bits per second, higher speeds are now used. To adjust for these higher speeds, the designers of these new protocols had to adjust other parameters of the Ethernet protocol to preserve the properties of the slot time and minimum message size described above.

Assuming the designers of the higher speed protocols were unwilling to change the slot time (T), what other parameters would have to be changed to make the collision detection and backoff protocols work correctly at the new, higher value of R, and would they be increased or decreased?

If the slot time is not changed then the minimum message length must increase given the formula shown in part b.

4) For each of the following program snippets, what is the final value of the variable result? Show the value of any intermediate expressions for partial credit.

a)

```
int result = 0;

int x = 5;

if (x > 4) {

    result = 1;

} else if (x > 3) {

    result = 2;

}
```

ANSWER: 1

b)

```
int result = 0;

int x = 5;

if (x > 4) {

    result = 1;

}

if (x > 3) {

    result = 2;

}
```

ANSWER: 2

c)

```
String result = “Talk to me”;

result.toLowerCase();

int pos = result.indexOf(“t”);

int end = result.length();

result = result.substring(pos,end);
```

ANSWER: “to me”

d)

```
String result = “Talk to me”;

result = result.toLowerCase();

int pos = result.indexOf(“t”);

int end = result.length();

result = result.substring(pos,end);
```

ANSWER: “talk to me”

e)

```
int result = 1;

while (result < 10) {

      result = result * 2 + 1;

}
```

ANSWER: 15

5) Below you will find two short phrases that use only the four letters A, E, R, and T (we will ignore the spaces between words in this question). In addition, below each phase you will find a table showing how often each of the four letters appears in the phrase and the probability that each of the four letters would be picked if a symbol from the phrase was chosen at random.

For this problem, we would like you to consider the process of sending each message using a separate Huffman code that is customized to minimize the bits required to transmit that message. Ignore the spaces when encoding the message; that is, assume that the letters are all run together when the message is sent and that the receiver must infer where words start and stop.

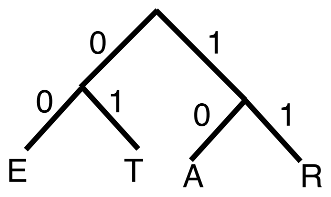
EAT A TART TREAT

Letter	Occurrences	Probability
E	2	0.16
A	4	0.30
R	2	0.16
T	5	0.38

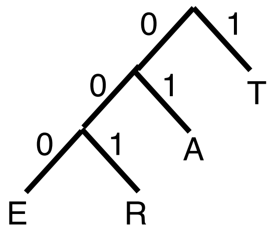
RETREAT AT A RARE RATE

Letter	Occurrences	Probability
E	4	0.22
A	5	0.28
R	5	0.28
T	4	0.22

a) Build a tree that describes a Huffman code for the message "RETREAT AT A RARE RATE".



b) A tree for encoding the phrase "EAT A TART TREAT" is shown below:



Given this tree, which binary code would be used for each letter?

E: 000 A: 01 R: 001 T: 1

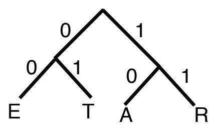
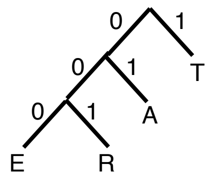
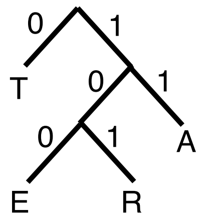
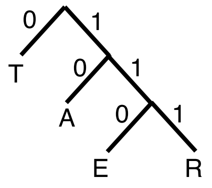
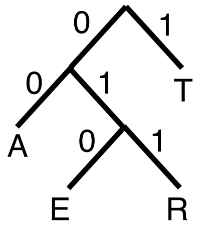
c) Which of these two messages achieves the best compression? That is, which code will require the fewest bits per letter to encode its message. Briefly justify your answer without actually encoding both messages. Do not count the cost of storing the tree in the size of the message.

The compression would be more effective with the message Eat a tart treat because the frequencies with which the letters appear in this message are less uniform than in the other message.

d) When messages that use only a few distinct symbols are being encoded, the number of possible Huffman codes is actually quite small. For example, if one tries to construct a Huffman code for a message encoded using only three symbols (a, b, and c?) there are actually only two sets of possible codewords:

1, 01, 00 and 0, 10, 11

For alphabets that use only four symbols (A, E, R, and T?), there are only five distinct sets of codewords that can make up a Huffman code. In the five columns of the table below, show the each of the sets of possible codewords and the Huffman tree that would correspond to the codewords in the set. Do not bother to label the edges of the tree with 0's and 1's. Just show the shapes.

	Set 1	Set 2	Set 3	Set 4	Set 5
Code words	*00* *01* *10* *11*	*000* *001* *01* *1*	*0* *100* *101* *11*	*0* *10* *110* *111*	*00* *010* *011* *1*
Tree	 0 1 0 1 0 1 E T A R	 0 1 0 1 T 0 1 A E R	 0 1 T 0 1 0 1 A E R	 0 1 T 0 1 A 0 1 E R	 0 1 0 1 T A 0 1 E R