Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Department of Mechanical Engineering

6.050J/2.110J **Information and Entropy** Spring 2003

Issued: February 4, 2003 **Due: February 7, 2003**

Problem Set 1

Laboratory Assignment

In 6.050J/2.110J, MATLAB will be used frequently in problem sets and demonstrations. MATLAB is a mathematics software package that is efficient at matrix operations as well as a good tool for data visualization. Access MATLAB on any Athena workstation by typing `add matlab` at your `athena%` prompt. Then type `matlab &` to enter the program. A separate window will appear, in which you can enter MATLAB commands.

Whenever a problem requires MATLAB, write a text file that holds the commands or functions that you would normally type into the MATLAB command line. This text file, also known as an M-file, should have a `.m` extension (e.g., `filename.m`). This file can be executed in the MATLAB window by typing the filename without the `.m` extension. This takes away the drudgery of retyping commands over and over again. Programs like vi, textedit, emacs, and pico can be used to create and edit the M-file on Athena. Moreover, we also request that you type `diary` (in MATLAB) which keeps track of all your commands and outputs from MATLAB. The results are placed in a file called diary. Please create a separate M-file for each problem and edit the diary for readability.

Go through the MATLAB tutorial given out in class to familiarize yourself with the syntax and environment. Remember that at any time, you can type `help` at the MATLAB prompt (before typing this, type `more on` to turn on page-by-page viewing). It is our intention that MATLAB be used as a helpful tool in this course, and that it not be a barrier to anyone's success in the course. We also hope that you'll develop skill with MATLAB which may be useful to you in other classes. To that end, we're here to help you if you run into problems, so feel free to ask us for assistance. If you have any questions, please email 6.050-staff@mit.edu.

Problem 1: Completely Logical

If A and B are boolean values, then it is alleged that

$$(\overline{A \cdot B}) + \overline{B} = \overline{A \cdot B}$$

where the overbar denotes logical negation (the NOT function). Using MATLAB, test whether the above law is true for all possible values of A and B. Hint: MATLAB has functions OR, AND, and NOT. These same functions can be applied using |, `&`, and $\sim$, respectively.

Problem 2: Universality

A Boolean function $F(A, B)$ is said to be **universal** if any arbitrary boolean function can be constructed by using nested $F(A, B)$ functions. A universal function is useful, since using it we can build any function we wish out of a single part. For example, when implementing boolean logic on a computer chip a universal function (called a 'gate' in logic-speak) can simplify design enormously. We would like to find a universal boolean function. In this problem we will denote the two boolean inputs A and B and the one boolean output as C.

a. First, to help us organize our thoughts, let's enumerate all of the functions we'd like to be able to construct. How many different possible one-output boolean functions of two variables are there? I.e., how many functions are there of the form $F(A, B) = C$?

b. Our first guess for a universal gate is OR. Is the $OR(A, B)$ function universal? Why or why not?

c. Next we might try AND. Is the $AND(A, B)$ function universal? Why or why not?

d. XOR is another common logic function. Is the $XOR(A, B)$ function universal? Why or why not?

e. Finally we will examine the $NAND$ function. Is the $NAND(A, B)$ function universal? Why or why not?

Prove your answers to parts b through e either by exhaustive enumeration of all cases (in MATLAB) or give a detailed explanation. Remember that you may use constants 1 and 0 as inputs as well. Write down your answers at the end of your diary.

Problem 3: Make it Fit

An important consideration in the design of codes is efficiency. In fixed-length codes like ASCII each symbol (in the case of text, each character is a symbol) is represented in the same number of bits. In variable-length codes the more frequently occurring symbols are assigned shorter codes and less frequently occurring symbols longer codes. If all symbols are equally likely there is no advantage to variable-length codes.

The Penny-Dreadful Book Company needs an efficient code suitable for everyday English text, one that is more efficient than ASCII (7 bits per character). The text they are interested in consists mainly of lower-case letters. Two of their engineers (one of whom, incidentally, was educated at Caltech, and the other at MIT) have each devised a code, and the company has asked you to advise them which to adopt.

In the first proposed code (designed by the Caltech graduate) lower-case letters are represented by 5-bit sequences. Since there are only 26 such letters, there are 6 unused sequences. One of these is assigned to 'space', and one to the control code 'ETX' signifying the end of the text. The code is defined so that the four unused codes are all of the form 111xx where each x is either 0 or 1. The remaining characters are encoded using 9 bits each, starting with 111 so that they cannot be confused with the codes for lower-case letters.

In the second proposed code (designed by the MIT graduate), lower-case letters, space, ETX, and four common punctuation marks (period, comma, question mark, and exclamation point) are represented by 6-bit sequences, all of which start with 0. All other characters are represented by 7-bit sequences which start with 1.

a. You need to represent ETX and all the ASCII characters except the control characters. This includes 26 lower-case letters, 26 upper-case letters, 10 digits, space, and 32 punctuation marks. Can both codes do this? Explain your answer.

b. If either code can not do this, see if you can modify the design slightly to permit all the characters to be represented.

c. Compare the number of bits required for a typical paragraph with 200 lower-case letters, 50 spaces, 20 upper-case, 5 common punctuation marks, 20 uncommon punctuation marks, and one ETX for whichever of Penny-Dreadful's codes that work, as well as ASCII and the typical 8-bit code used in computers.

d. It is pointed out that there is also need for some specialized control characters, such as new-paragraph and new-chapter, and additional punctuation marks such as em-dash, en-dash, and opening and closing curly single and double quotes. Describe in words how you would propose extending the second proposed code to allow 12 additional characters.

e. **Extra Credit:** Implement an encoder and decoder for the MIT graduate's code. For simplicity only include the first seven letters, space, period, exclamation point, and ETX. Apply it to the phrase "Egad! Deb bagged a cad." Using MATLAB is recommended, but not required.

Turning in Your Solutions

You should have three files (two M-files and one diary). Name the M-files `ps1p1.m` and `ps1p2.m` and name the diary `ps1diary`. You can rename your files at your `athena%` prompt using the following command.

```
mv oldfilename newfilename
```

Turn in this problem set by e-mailing your M-files and diary along with your answers to any problem(s) not done using MATLAB, to 6.050-submit@mit.edu. You may do this either by attaching them to the e-mail as text files, or by pasting their content directly into the body of the e-mail (if you do this, please somehow indicate where each file begins and ends). Alternatively, you may turn in your solutions on paper in room 38-344. The deadline for submission is the same no matter which option you choose.

Your solutions are due 5:00 PM on Friday, February 7, 2003. Later that day solutions will be posted on the web.