Comments on the basic elements of computer languages

All computer languages include the same basic elements: (1) input and output, (2) data storage, (3) processing, (4) decisions, (5) repeats, (6) procedures.

Only six elements? How can computers do so many complex things with only six elements? They can because (a) decision and repeat structures can be nested inside each other and procedures can "call" themselves and other procedures, and (b) nonlinear theory has proven that the interaction of only a couple simple elements can result in extremely complex behavior - a big idea!

Here are some notes on these elements, especially as they relate to a loan calculator students wrote one year in the course.

1) (a) Input

For input in the loan calculator, we suggested the `input` function, which is a standard Matlab function (see procedures below). Here is an example:

```
amt = input('enter the amount financed: ');
```

This is an "assignment statement." On the left is the name of a variable. The equal sign is the assignment operator. On the right is an expression. The expression is evaluated to produce a value, then the value is assigned to the variable on the left.

In this case, the expression on the right is a reference to the standard Matlab function `input`. In parentheses is the value of an input "argument" required by the function. Apostrophes enclose a string (of text characters) value. `Input` will display this in the command window and then wait for the user to enter a value from the keyboard. Then `input` returns this value to the program, and it becomes the value of the expression on the right.

1) (b) Output

For output in the loan calculator, we want to see monetary values with two digits to the right of the decimal point. The quick way is to use the command `format bank`. This switches (toggles) the output format in the command window, which will stay this way until you switch it again.

Another way is to use the standard Matlab function `fprintf`, i.e., file print format, where the "file" here is the screen.

```
fprintf('monthly payment $ %6.2f \n', pymt);
```

Here, `fprintf` has two input arguments: a string and a variable. The string is displayed in the command window with the exceptions of `%6.2f` and `\n`. In the place of the format specification `%6.2f` the value of the variable `pymt` will appear. The `\n` starts a new line in the command window.

The format specification starts with %. The 6 says that a minimum of 6 spaces in the output will be used to display the value, counting the decimal point. The 2 says that 2 digits will be displayed to the right of the decimal point. Experiment with changing these numbers to see what happens. The format specification ends with f, which means to display a floating-point number.

2) Data Storage

There are several different types of data that can be stored by Matlab, including floating point numbers (single and double precision), integers (signed and unsigned of bit lengths 8-64), arrays of alpha-numeric characters called "strings" (string of characters), and Boolean (logical).

Data you wish to store can be assigned to a variable name in an assignment statement, which is a variable name followed by an equals sign followed by an expression. For example, `a = 5`, or `a = 2*b`.

There are four characteristics of a variable: name, type, location in memory, and values of the data. Whenever Matlab encounters a variable name in an expression, it retrieves the data from the location where the data is stored in memory and uses the values of the data to evaluate the expression.

All data types are stored in memory as binary numbers. All data is converted to binary numbers in input devices in order to store the data in the form of the binary (either-or) states of a physical memory storage device. Examples of physical data storage devices include transistors in integrated circuits (on-off), capacitors (charged-discharged), field orientation of magnetic elements in hard disks (N-S), and optical reflectivity of organic dies or aluminum films in DVDs (specular-

diffuse). Within a digital computer, and within digital communication systems, all data is in binary form. Data is converted from binary back to another form in output devices.

The default numeric type in Matlab is double-precision floating point. Double-precision means that 64 binary digits (bits, transistors) are used to store a numeric value. Floating point means a non-integer number. The first bit in memory is used to store the sign, − or +. Eleven of the bits are used to store the exponent or power of ten of the number when expressed with one digit to the left of the decimal and a power of ten, e.g., `+5.000e+00`. The remaining 52 bits are used to store the number part of that expression, the significand.

Because of the fixed number of bits used to store the exponent and significand, there are limitations to both the magnitude and precision of data that can be stored. Thus, there are limitations to the magnitude and precision of results produced by numerical calculations.

Alphanumeric characters are converted to binary numbers via the process of a table of conversions. Conversion table systems include ASCII and several varieties of Unicode. For example, "A" is first converted in ASCII to the decimal number 65 and then to the binary equivalent of 65, which is 01000001. Bits are added to denote that the value is a character value. ("a" is 97)

3) Processing

You did some math processing already in HW 1. Learn the basic operators `(+ - * / ^)` and the order of precedence (order of evaluation) and how that can be controlled with parentheses.

4) Decisions

The basic decision structure in Matlab is the "if" structure. If the value of a logical expression is true, then the lines of instructions between `if` and `end` are executed. The value of a logical expression can be true (not zero) or false (zero). The logical expression `(5 == 5)` is true, where double equal signs are the logical equivalence operator, and a single equal sign is the assignment operator.

We will learn more about decisions later.

5) Repeats

If you want to do something simple like compute a student's GPA, you can write some simple code (script, instructions). If you want to compute the GPA's of every student at UCSD, the main thing you have to do is put the code for one student inside a repeat structure and repeat the same calculation for everyone.

There are two types of repeat structures in Matlab: "for" (when you already know how many times to repeat) and "while" (when you don't). Use while when you want to let your user compute several loan cases.

```
while (logical value)
  % while the logical value is true...
  %       repeat executing the code between while and end
end
```

For example, I suggested

```
reply = 1; % "initialize" the value of reply, where 1 is true
while reply
  % as long as reply is not 0 (zero, false) we get here
  % insert loan calculations here
  fprintf('inside while structure \n') % just to see what happens here
  reply = input('enter 0 (zero) to quit, 1 to repeat: ');
end
```

6) Procedures

When you write some useful code that you want to use again, you don't want to copy and paste it. Why not? Because if you want to change something, you have to change it in all your copies.

Instead, package your code into a procedure. In Matlab, the procedures we use are called functions. You can use the same function many different places. When you want to make a change, you only have to change the one copy of your function.

Matlab has many standard or built-in functions. An example is `input`. Functions are defined in m-files. Somewhere in the Matlab distribution is an m-file named input.m. When you type "help input" in the command window, the first set of comments in the input.m file are echoed to the command window.

We will learn to write our own functions.

Matlab, Fortran and C are called "procedural languages" because instructions are executed according to the procedure first to last.

There are other languages like C++ and Java that are called "object-oriented languages." Objects are code structures that can contain data as well as instructions and send messages to each other. In my opinion, the "objects" in such a language can be considered as fancy procedures. As you learn more about Matlab, you will learn that it can also use objects.

Your goal is to learn how to solve problems using Matlab, not simply to complete homework assignments. As one step to achieving this goal, I recommend writing short "test" m-files and test lines in the command window. Don't stick to only working on your homework m-file. For example, to learn about the "while" repeat, make a separate m-file in which you can experiment. After you understand what is going on, then incorporate it into your homework m-file.