Homework 7

This homework is Friday April 6 (by 5:00 PM). Like all other assignments, this homework is to be done individually.

Please mail your completed programs to mobasher on Zippy by executing the following commands in the directory where you have done your CLU development.

```
shar -m Makefile *.equ *.spc *.clu simulate | mailx -s "hw7 solution from ME"
```

Also turn in a listing of your program, consisting of your code (and the Makefile, if you changed it and any specification or equate files that you may have changed or added.)

1. (10 points) Consider handling exceptional situations by passing procedures as arguments. For example, instead of writing the pop operation of a stack as follows

```
% pop with standard exception handling
pop = proc(s: stack) signals(empty)
    if empty(s)
    then
        signal empty
    end
    % ...
end pop
```

one would write the following:

```
% pop with exceptions handled by procedure arguments (the alternative)
pop = proc(s: stack, error: proctype())
    if empty(s)
    then
        error()
        return
    end
    % ...
end pop
```

As another example, consider the top for stacks. If exceptional situations are handled by passing procedures as arguments, then the top operation would be written as follows.

```
% push with exceptions handled by procedure arguments (the alternative)
top = proc(s: stack, error: proctype() returns(element)) returns(element)
    if empty(s)
    then
        return(error())
    end
    % ...
end top
```

What are the advantages and disadvantages of using CLU's exception handling mechanism as compared with the alternative?

2. The exception "failure" is treated specially by CLU. For example, "failure" cannot be declared as an exception in a procedure heading, although it is possible to raise a "failure" exception from within any procedure.
   (a) (5 points) What are the advantages and disadvantages of CLU's treatment of "failure"?
   (b) (10 points) Suggest a revision to the CLU exception handling mechanism that would eliminate the special treatment of "failure" and any disadvantages you identify.

3. CLU does not provide a built-in type that corresponds to subrange types in Pascal or Ada (see MacLennan, pages 185 and 273).
   (a) (10 points) Write a CLU cluster that has roughly same behavior and error checking (security) properties as the subrange type mechanism of Pascal or Ada.
   (b) (5 points) Was it a good idea to not have subrange types in CLU?

4. (45 points) Implement the Turing machine simulator described in the *Turing Machine Reference Manual.* The simulator should take its program and input from the Unix standard input.

   You should use the design described in class as the basis for your program. (You are required to use all the clusters we specified, but feel free to add more if you wish.) To do this, execute the following commands

   ```
   mkdir clu
   cp /usr1/users/class/cs342/public/clu/turing-machine/* clu
   ```

   which will copy the files that describe the design from the course directory to your account into a subdirectory called `clu`. Read the file named README in that directory to get started. Your task is to provide code for all the procedures and clusters; for the clusters you will also have to choose a representation.