APS 105S – Computer Fundamentals

Assignment #4: Recursion

Fall 1996

(To be completed in your lab period Oct. 17/18)

Objective: To implement recursion in Turing programs to perform the calculation of the factorial function of Assignment 3, to implement a recursive function, and to obtain, store and print out a list of words.

1 Factorial Function

Recall in Assignment 3 you implemented a calculation of the factorial of a given number. This calculation can also be implemented recursively. A Turing program has been created to allow you to trace recursive calls as they occur in the evaluation of a factorial. Copy the contents of file

`/usr/copy/aps105/a4A.t`

to a file called `factorial.t` in your own directory. Call up this program in oot, study it, and test it with various input values. You will discover that you can compute factorials only up to a certain value. Determine this unlucky limit, and be able to explain why higher factorials cause the program to fail.

2 Recursive Function

Design, implement and test a Turing program to calculate a series of numbers given the following function. You may use the program from Part 1 as a model. Your program should trace each call to the function (entry & exit), and print the total number of function calls made. The function is defined as follows:

$$\begin{aligned} F(x) &= F(x-1) + 1/10x \qquad && \text{for } x \geq 1 \\ &= 0 && \text{for } x = 0 \end{aligned}$$

This recursive function is used to solve the following word problem (provided for interest only). Using the divide and conquer method and step-wise refinement, justify to yourself the correctness of the recursive function used to calculate the distance travelled.

A worm crawls along a 1-meter rubber band at a constant speed of 10 centimetres per minute. After each minute, its keeper stretches the rubber band by an additional metre. Because the relative position of the worm does not change when the band stretches, after the first minute the worm is 1/10 of the way along a 2-meter band. During the second minute, it crawls another 10 cm making 30 cm out of 2 m or 3/20 = 1/10(1 + 1/2). After another 1 m stretch, the fraction represents 45 cm out of 3 m. (WHY?)

Let *f(n-1)* be the fraction of the band crawled after *n-1* minutes. During the *n*-th minute the worm crawls 10 cm while the band remains 100n centimetres long. Thus at the end of *n* minutes, the fraction is *f(n)* = *f(n-1)* + 10/(100n) = *f(n-1)* + 1/(10n).

3 Printing A Wordlist in Reverse Order

Design, implement and test a Turing program which can read a list of words and print them in reverse order. The word "end" will end the list, and is not to be included in the output. You are to use recursion to do this rather than a data structure (i.e. no arrays or collections). Hint: Design a procedure "getmore" that reads a word (a **string** in Turing), returns if the word is "end", and otherwise calls itself (to process the other words in the list) and then prints the word. Your code will look something like this.

```
procedure getmore
        var s: string
        put "Enter word: "..
        get s
        .
        .
        finish this code
        .
        .
end getmore

put "Enter a list of words, ending with \"end\"."
getmore
```

The execution of the program will produce something like this. The input supplied by the user is shown in **bold**.

```
Enter a list of words, ending with "end".

Enter word: recursion
Enter word: is
Enter word: useful
Enter word: end
useful
is
recursion
```

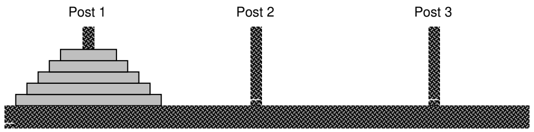

Figure 1: The Towers of Hanoi

4 Optional Section — Towers of Hanoi

The Towers of Hanoi game is the classic problem suited to a recursion programming solution. The set-up of the problem is illustrated in the Figure 1.

There are three upright posts in a line that are named post 1, post 2, and post 3. A number of rings of different sizes could be stacked up around the posts. To start, all of the rings are stacked on post 1, in order of decreasing size from the bottom up. The object of the game is to relocate the entire stack of rings from post 1 to post 2. The movement of rings is governed by two rules: 1. only one ring can be moved at any time from one of the posts to any of the other posts; and 2. at no time can a ring be placed on top of another ring of smaller size.

Determine the algorithm required to solve the Towers of Hanoi problem and implement it in a Turing program. The program should accept from the user the number of rings in the stack, and the program will list the directions to solve the puzzle. Here is an example of the output of the program:

```
        TOWERS OF HANOI
How many rings are in the stack?  3

Solution:
Move ring from post 1 to post 2.
Move ring from post 1 to post 3.
Move ring from post 2 to post 3.
Move ring from post 1 to post 2.
Move ring from post 3 to post 1.
Move ring from post 3 to post 2.
Move ring from post 1 to post 2.
```

You can verify that this is the solution for a stack of three rings.