APS 105S – Computer Fundamentals

Assignment #2: Selection and Iteration

Fall 1996

(To be completed in your lab period Sept. 26/27)

Objective: To devise a combinatorial solver that calculates C(n,r), the number of ways to choose **r** objects from a set of **n** objects. This will require the use of IF statements to detect potential errors and LOOP states for repetition.

1 Calculate C(n,r)

In the lectures we have seen examples of logic errors and the use of IF statements to detect the conditions leading up to a logic error. We have also seen the use of repetition to accomplish a multiple-step task. In this lab we will combine these two constructs and calculate "**n** choose **r**".

Write a simple Turing program that solves the problem described below and call the program choose.t . Obtain the values of the variables **n** and **r** from the user. With these values, calculate

$$C\binom{n}{r} = \frac{n!}{(n-r)!r!}$$

Your program should work for all values of **n** and **r** and detect all special cases, including **n** or **r** less than zero, and **r** greater than **n**. A solution or explanation should be provided for all cases.

HINT: n! is a classic case for the use of repetition. Determine an algorithm to calculate factorials and **then** determine an algorithm for **n** choose **r**.

2 Processing Several Sets Of Data In One Run

The program that you have devised will determine the solution of **n** choose **r** for only one set of values. If you want to calculate the number of combinations of more than one set of values, you must run the program more than once. A common technique used in computer algorithms is the repetition of a sequence of steps. A construct in Turing that allows this is the loop. The following program illustrates this:


```
var a: string
loop
    put "Would you like to continue?"
    get a
    exit when a = "no"
end loop
```

Type in this program and run it. You will see that the user is asked "Would you like to continue?" and a single word is expected as a reply. If the word entered is "no" then the loop is exited and execution of the program terminates. Otherwise, execution loops back to the put statement and the sequence of statements is repeated. Repetition of this sequence of statements continues until the user responds with "no". Run this program again, trying "NO", and "No" as responses.

Using this simple program as a model, encase your combinatorial calculator program inside a loop. Your program should always begin by asking for (n,r) and then respond either with the number of possible combinations or a message, and then ask the user if s/he wishes to solve another set of values.

3 Optional Section Input/Output Redirection

The programs you have designed so far have input data from the user at the computer terminal and have output text to the screen. For this exercise, you will experiment with reading in data from a text file and writing output to a text file. The Turing statements you will have to reference to complete this exercise are **open**, **close**, **get**, **put**, and **eof**.

Write a new version of your combinatorial calculator program (called choose2.t) that reads in values of **n** and **r** from a file names "chooseData.txt". This file (which you must first create) will contain one or more rows of data, each row having two numbers corresponding to **n** and **r**. A sample chooseData.txt file might look like this:

```
 10    2
  2    1
191  -15
```

Your program is to read in the data from the file, one row at a time, solve the combinatorial calculation then display the results on the screen. The program should repeat these steps until all rows of data have been read. Display your results in tabular form to make the results clear. For the sample chooseData.txt file shown above, your program should produce output that looks something like the following:

```
            Combinatorial Calculator

Solutions are given for the combinatorial equation,
n! / ( (n-r)! r! ) with the coefficients listed below:

     n        r    C(n,r)
-----------------------------
     10       2      45
      2       1       2
    191     -15      Error: Can't choose a negative number of objects
```

Modify your program so that it sends the output to another text file, "choices.txt". This modification should require only a few small changes.

4 Optional Section: UNIX I/O Redirection

Using **toot**, generate an executable version of choose.t (choose.x). Use UNIX I/O redirection to read data from ChooseData.txt and / or put the output into choices.txt.