APS105F: Computer Fundamentals
Lab Quiz, October 10, 1996

Instructions

- Place your student card on the desk top, so that it can be checked by the examiner.
- The only aid allowed is an original edition of the "Turing Reference Manual", the "On-line Turing Reference Manual", or other book that has a manual for Turing. You may not bring or use a photocopy of that manual, any other books, notes, written materials, or a calculator. You may reuse any programs in your own directories in building your program for this quiz.
- You must not communicate in any way (oral, written, or electronic) with anyone other than the examiner and his or her assistants. You should only talk to them if you believe there is an error or other problem in the program specifications. They will not give you any advice.
- There are three parts to this exam. The answer to each part will consist of a Turing program which satisfies the specifications given below. You will compose these programs in files called "`part1.t`", "`part2.t`", and "`part3.t`", respectively.
- **Insert your name and student number as comments at the beginning of each file.**
- You will submit these programs for grading by typing the commands "`SUBMITPART1`", "`SUBMITPART2`", and "`SUBMITPART3`", respectively. Note that these commands are Unix commands and they are typed as one word (no blanks) in block capitals. If you are in "oot", you must save your Turing file and exit to UNIX before running this command. The command will check that you have an appropriate file in your current directory.
- You may submit revised versions of your programs at any time before the end of the quiz by again typing the submit command. However, if you run this command before your quiz starts, or after the examiner has announced your quiz is over, you will receive a grade of zero.
- You may use the back of the pages for your rough work and pseudo-code design.
- The examiner will announce when there are 10 minutes and 5 minutes remaining in the quiz. It is your responsibility to submit your program before the end of the two hour quiz period. It is strongly advised that you do not leave this submission to the last minute.
- It is highly recommended that you save the file you are working on periodically; i.e. every 10 minutes or so.
- Complete and sign this sheet, and hand it in at the end of the quiz.

Name (print)______________________________

Login ID (print)______________________________

Student Number______________________________

Signature______________________________

Marks:	1 [5]		2 [8]		3 [2]		Total [15]	

1 Summing a Sequence

Write a program that starts by requesting an integer in the range $0 < N < 20$. Numbers outside this range are rejected and a new request is made. Output the sum of the sequence of numbers starting at 1 and ending with N. The program then exits. Use the format shown in the example below:

With $N = 5$, the output will be

```
1 + 2 + 3 + 4 + 5 = 15
```

Restrictions:

You may **not** use a `for` statement for this program, even though it probably makes more sense to do so. Also, do not use `locate` to position the output.

Hint:

You can use the Turing command "`put x..`" (as opposed to "`put x`") to prevent the cursor from going on to the next line after the value of `x` is output.

2 GIC Interest

A bank wants to show its customers how their savings will grow if they invest in a Guaranteed Investment Certificate (GIC). It is assumed that the customer starts with $1000. The bank's GIC plan has three interest rates, depending on the number of years that the GIC is held.

Years	Rate
1	Rate1
2	Rate1
3	Rate2
4	Rate2
5 or more	Rate3

The file `/usr/copy/aps105/gicrates.t` contains the code that does most of this. There are three modules that do not exist: `GetRates`, `WhichRate`, and `Compute`. The `GetRates` module gets the three rates as input. You may assume that only valid inputs will be provided. Assume that these rates will be integers. The `WhichRate` module returns the appropriate interest rate based on the term (years) of the GIC and `Compute` returns the final GIC value given a rate and the number of years. For N years and a rate of $r\%$, the final value is

$$1000 * (1 + r/100)^N = 1000 * (1 + r/100) * *N$$

The code in `gicrates.t` reads as follows:

```
loop
    var rate1,rate2,rate3,rate : int
    var years : int
    var finalvalue : real

    GetRates(rate1,rate2,rate3)
    exit when rate1 = 0

    loop
        put "Years? " ..
        get years
        exit when years = 0
        rate := WhichRate(rate1,rate2,rate3,years)
        finalvalue := Compute(rate,years)
        put "After ",years," years you will get ",
            finalvalue:9:2, " dollars."
    end loop

end loop
```

You may include the code of `gicrates.t` either by copying from the file

`/usr/copy/aps105/gicrates.t`

into your file "`part2.t`" or by simply having the statement

`include "/usr/copy/aps105/gicrates.t"`

after the declaration of your modules.

Restrictions:

The modules you write may not use any global variables and they are not to output anything. You may not change the code you import from `gicrates.t` in any way.

3 A Sequence Recognizer

Note: The difficulty of this question is not in proportion to the grades assigned. Make sure you do the other questions first!

Write a program that receives as input a sequence of ones (1) and zeros (0). Any number that is not one or zero will end the program. For every input digit, there is an output digit. A one is output when the most recent input digit completes the pattern "1101" (i.e., 1, then 1, then 0, and then 1 is input), otherwise a zero is output. Overlapping patterns are not recognized. An example sequence is shown below. The first input is at the left end.

```
Input:  0 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1 0 0 ...
Output: 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 ...
```

Restrictions:

You should not do any fancy input and output formatting. Just read an input and output 1 or 0 with a simple `put 1` or `put 0` statement. Assume that all inputs will be integers.

Please:

- **log on to the system**
- **start up oot**
- **do not turn this over until you are told to do so.**