

Program 2: Circle in FORTH

CS 636 / 338 Spring, 2002

Goals

- To write a very simple program in FORTH.
- To practice writing code in a highly modular style.
- To learn to do stack manipulation, arithmetic, and I/O in FORTH.
- To write at least one function with parameters and learn how to return a value from a function.
- To design adequate, well chosen test data.
- To learn something about integer arithmetic.

Overview. Write a program that will read a series of inputs from the stack. Each number represents the diameter of a circle. If the input is 0, terminate the process. If the input is negative, print an error comment. If the input is positive, print out a table like this:

```
Diameter = 200
Radius = 100
Circumference = 628
Area = 31416
```

Use 2-byte integer arithmetic. Integer arithmetic on a computer has tricky semantics, and the order of operations in a formula does matter greatly. Note that you cannot define PI separately as a constant, because it is too inaccurate to be useful. The following two FORTH definitions are equivalent and both are unacceptable:

```
3 constant pi
22 7 / constant pi    ( 22/7 should be=3.142857 but is=3 )
```

Since you cannot pre-compute PI, you must multiply by 22 and divide by 7 somewhere in your formula. The accuracy of the result and the maximum input you can handle depend on where you put the division in the formula. Some possible formulas (not in FORTH notation) for the area are:

1. $area = r * r * PI$
2. $area = d * d * PI / 4$
3. $area = 22 * d * d / 28$
4. $area = *(d, 22, 7) * d / 4$

It does make a difference which formula you use. If you divide early in the sequence of operations, the round-off error will be compounded by later multiplications. For example, if you use formula (1), above, inputs 10 and 11 will produce the same output! On the other hand, if you use formula (3), the intermediate results from relatively small inputs will grow too big to be represented in two bytes, resulting in arithmetic overflow.

There is no single “right” order in which to compute the area. Some formulas are clearly bad. Most, however, have both bad and good properties. In any given application, a programmer must choose an appropriate version. For this problem, choose a formula that gives different results for diameters of 10 and 11. Then use your program to experimentally determine the largest input which does not cause arithmetic overflow.

Multiply, then divide. The FORTH operator “*/” is unusual and useful. This operator takes three 2-byte operands from the stack. It multiplies the first two together, giving a 4- byte intermediate result, which is then divided by the third operand, producing a 2-byte answer.

Hand in a test plan, a listing of your program and a transcript of the test of your program.