CS 302
Elaine Rich

Algorithms

1. Consider the problem of making a peanut butter and jelly sandwich.
   a. Write an algorithm to accomplish the task.
   b. Mark the places where nontrivial knowledge is required and indicate what it is.

2. In class, we showed the circuitry for binary addition. Of course, in real computers, we need to perform other arithmetic operations as well. Let's consider multiplication. Notice the relationship between multiplication and addition:

   $5 * 3 = 3 + 3 + 3 + 3 + 3$

   Imagine that you have a machine with an addition circuit but you now want to do multiplication. You can do it in software. Fill in the required code in the following Python function. (Hint: actually type this up and run it. That way you can be sure you submit a correct answer.)

```
     def mult(n, m):
         """ Multiplies n by m, using only addition.
             Precondition: n and m are positive
integers.  """

         # code to compute n * m and make it the value
of the variable ans



         return(ans)
```

3. Consider the following chunk of code that we used as an example of the `if/elif/else` construct:

```
     if x < 10:
          print("small")
     elif x > 100:
          print("big")
     else:
          print("so-so")
```

   This code won't work in all situations. What must be true before it starts in order for it not to produce an error? In other words, what is its precondition? (Hint: Run it on various inputs. Think about things it isn't meant for.)

4. Recall this program that we looked at when we were learning Python.

```
def vowels(st):
    new = ""
    for i in range(len(st)):
        if st[i]=="a" or st[i]=="e" or st[i]=="i" or
st[i]=="o" or st[i]=="u" or st[i]=="y":
            new = new + "*"
        else:
            new = new + st[i]
    return(new)
```

   a. Describe in English what it does.
   b. Use big-O notation to describe how many steps it executes as a function of |st| (the length of the input string st).

5. What is the sum of the integers from 1 to 9876?

6. The prime factorization of a positive integer n is the bag (think of a bag as a set but duplicates are allowed) of positive integers with two properties:

   - The product of the integers in the bag is n.
   - All the integers in the bag are prime. (1 is not a prime number, so cannot be in the bag.)

   So, for example, the prime factorization of 612 is {2, 2, 3, 3, 17}. The prime factorization of 17 is {17}. What is the prime factorization of 714?

7. Recall the code we wrote to implement Euclid's method for determining the greatest common divisor (gcd) of two integers:

```
def gcd_euclid(n,m):
if m == 0:
        return(n)
    else:
        return gcd_euclid(m, n%m)
```

   Compute gcd(87, 6). Use `gcd_euclid` and show each of the calls to it starting with `gcd_euclid(87,6)`.

8. In class, I presented the function `newton` that computes square root using Newton's method:

```
def newton(n):
    guess = 1
    while abs(guess**2 - n) > .00000001:
        guess = (guess + n/guess)/2
    return(guess)
```

Use it to find sqrt(1287658). It's fine to get it accurate to two decimal places.

Note that sqrt is the name that mathematicians use to describe the square root. The function we have just defined is called `newton`. So you will want to invoke it by typing `newton(1287658).`

9. In class we looked at this applet for experimenting with the Towers of Hanoi problem

http://www.mazeworks.com/hanoi/index.htm

Start it up. Give it four disks. Call the leftmost pole 1, the middle one 2, and the rightmost one 3. Solve the problem (i.e., move the pile of disks from pole 1 to pole 3. Show the sequence of moves you make. You can indicate a move by writing just the number of the pole that you are moving a disk to. How close did you come to solving the problem in the optimum number of moves?