CS 190C: Introduction to Computational Thinking

EXAM 1

WEDNESDAY, FEBRUARY 13, 2008

1.) (i) (6 pts) Given that $2^8 = 256$, and suppose that a computer uses 8 bits to store signed integers, answer the following questions.

1. How many different integer values can be represented? 256
2. What is the largest positive integer that can be represented? 127
3. What is the smallest negative integer that can be represented? -128

(ii) (10 pts) What is the value of each expression? Explain your answers.

```
>>> "hello"[2]
'l'

>>> "hello"[0:5]
'hello'

>>> "hello"[1:4]
'ell'

>>> [2/3, 2%3, 2**3]
[0, 2, 8]

>>> [5 + 5, "5" + "5"]
[10, '55']
```

2.) (10 pts) Which list does not have five elements? Give the value of each expression.

```
 >>> range(5)
[0, 1, 2, 3, 4]

>>> range(1,6)
[1, 2, 3, 4, 5]

>>> range(10,100,20)
```

```
[10, 30, 50, 70, 90]

>>> range(1,5)
[1, 2, 3, 4]

>>> range(10,0,-2)
[10, 8, 6, 4, 2]
```

3.) (15 pts) Which loops will terminate? Explain your answers.

```
(a)     i = 7
        j = 3
        while j > 0:
            i = i - 1
```

will not terminate as the value of j is not changed in the loop

```
(b)     i = 7
        while i <= 2 or i > 3:
            i = i - 1
```

will terminate when i=3

```
(c)     X = 0.0
        h = 0.1
        while X < 1.0:
            # add point (X,X**3) to a VPython plot
            graph.plot(pos=(X, X**3))
```

will not terminate as X is not changed inside the loop

4.) (20=6+8+6 pts) What will each program print?

```
(a) def fun1(j):
        j[0] = 2*j[0]

    a = ["a", 2, 3.1]
    fun1(a)
    print a

    a[0] = 5
    fun1(a)
    print a
```

```
['aa', 2, 3.1]
[10, 2, 3.1]
actually print 3.1000000000000001
```

```
(b)
def fun2(a, b = 6, cond = False):
    if cond:
        mid1 = (a+b)/2
        mid2 = float(mid1)
        print mid1, mid2
        return(mid1, mid2)

num1 = 3
print fun2(num1, cond=True)
num2 = 3.3
print fun2(num2, 8.0)
```

```
4 4.0
(4, 4.0)
None
```

```
 (c)  def fun3 (x, y):
          x = 25
          a = 2*x
          print "y before: ", y
```

```
        y[0] = y[0] + a
        print "y after: ", y

    a = 70
    b = [45]
    fun3(a,b)
    print a, b
```

```
y before:  [45]
y after:  [95]
70 [95]
```

5.) (12 pts) Function `recPower(a, n)` computes a^n using recursion. It makes use of the fact that for even n, $a^n = a^n/2 * a^n/2$. Knowing $a^{n/2}$ allows one to compute a^n with one multiplication. Odd values of n require one more multiplication.

```
def recPower(a, n):
# raises a to the n-th power
    print "n =", n
    if n ==  0:
        return 1
    else:
        factor = recPower(a, n/2)
        if n%2 == 0:        # n is even
            return factor*factor
        else:               # n is odd
            return factor*factor*a
```

1. What is printed for recPower(2,25)?

   ```
   n = 25
   n = 12
   n = 6
   n = 3
   n = 1
   n = 0
   ```

2. What is the total number of multiplications performed when computing 2^{25}.
   8 multiplications (if there were a second base case for n=1, there would only be 6)

6.) (15 pts) Below is an incomplete piece of code that detects horizontal percolation in a square grid (represented as a list of lists, as seen in class).

Augment the skeleton code so that it is a function `horiz_3row_perc(grid)` which returns True if there exist at least three rows that percolate and False otherwise. You can remove code. Indicate changes in indentation by making appropriate lines.

```
def horiz_3row_perc(grid):
    n = len(grid[0])
    count_3perc = 0

    for i in range(0,n):
        row_perc = True # true if current row can percolate
        for j in range(0,n):
            if grid[i][j] == 1:
                row_perc = False

        if row_perc:
           count_3perc = count_3perc + 1
           if count_3perc > 2:
               return True
    return False
```

7.) (12 pts) Below is the code of function `subreplace(string1, string2, m, n)` that replaces the characters in positions string1[m] to string1[n-1] with string2.

```
def subreplace(string1, string2, m, n):
    # recall that + stands for the concatenation of strings
    L = len(string1)
    if m < 0 or m > L:
        return string1
    if n < m or n > L:
        return string1
    S = string1[0:m] + string2 + string1[n:]
    return S
```

(a) What will each of the print statement print? Be careful to show all spaces printed.

```
T = subreplace("composition", "et", 4, 6)
print T
competition

T = subreplace("must go", " really", 4, 4)
print T
must really go

T = subreplace("it is time", "too late", 6, 4)
print T
it is time
```

(b) Complete the statement changing 'portable' into 'reliable':

```
subreplace("portable", "reli", 0, 4)
subreplace("portable", "reliable", 0, 8)
```