6.00 Introduction to Computer Science and Programming
Fall 2008

1) Is each of the following True or False

1.1. A greedy algorithm can be used to solve the 0-1 knapsack optimization problem.

1.2. Dynamic programming can be used to solve optimization problems where the size of the space of possible solutions is exponentially large.

1.3. Dynamic programming can be used to find an approximate solution to an optimization problem, but cannot be used to find a solution that is guaranteed to be optimal.

1.4. In Python, an instance of a class is an object, but a class itself is not.

1.5. In Python, a subclass can override a function definition contained in a super class.

1.6. Decision trees are <u>always</u> binary.

2) Provide code implementing a Python function that meets the specification below.

```
def findMedian(L):
   """Finds median of L.
   L: a non-empty list of floats
   Returns:
         If L has an odd number of elements, returns the median
         element of L.  For example, if L is the list
         [15.0, 5.3, 18.2], returns 15.0.

         If L has an even number of elements, returns the average
         of the two median elements. For example, if L is the
         list [1.0, 2.0, 3.0, 4.0], returns 2.5.

         If the list is empty, raises a ValueError exception.
   Side effects: none.
   """
```

3) What does the following code print?

```
class Shape(object):
    def __cmp__(s1, s2):
        return cmp(s1.area(), s2.area())

class Square(Shape):
    def __init__(self, h):
        self.side = float(h)
    def area(self):
        return self.side**2
    def __str__(self):
        return 'Square with side ' + str(self.side)

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3.14159*(self.radius**2)
    def __str__(self):
        return 'Circle with radius ' + str(self.radius)

def f(L):
    if len(L) == 0: return None
    x = L[0]
    for s in L:
        if s >= x:
            x = s
    return x

s = Square(4)
print s.area()
L = []
shapes = {0:Circle, 1: Square}
for i in range(10):
    L.append(shapes[i%2](i))
print L[4]
print f(L)
```

4) The following two formulas can be used to formalize a 0-1 knapsack problem.

1) $\sum_{i=1}^{n} p_i x_i$ **2)** $\sum_{i=1}^{n} w_i x_i \leq C$

4.1) What do each of n, p_i, x_i, w_i, and C represent?

4.2) Use the formulas (refer to them as "formula 1" and "formula 2") to describe the optimization problem to be solved.

5. Write pseudo code describing merge sort.

6) Consider the two functions specified below that are used to play a “guess a number game.”

```
def cmpGuess(guess):
    """Assumes that guess is an integer in range(maxVal). returns -1 if guess is < than the magic number, 0 if it is equal to the magic number and 1 if it is greater than the magic number."""

def findNumber(maxVal):
    """Assumes that maxVal is a positive integer. Returns a number, num, such that cmpGuess(num) == 0."""
```

Write a Python implementation of findNumber that guesses the magic number defined by cmpGuess. Your program should have the lowest time complexity possible. (20 points)