Problem 1

```
def output_digits_most(n):
    if n > 0:
        output_digits_most(n // 10)
        print(n % 10)
```

Problem 2

```
def make_checker(value):
    def checker(other):
        return other == value
    return checker

def make_reporter(value):
    def reporter(other):
        if other < value:
            print("smaller")
        if other > value:
            print("larger")
    return reporter
```

Problem 3

```
1045
1047
1097
[4, 1045, 100] [4, 1045, 100] [9, 1097, 1007]
```

Problem 4

```
class Counter:
    def __init__(self,start):
        self.count = start
    def increment(self):
        self.count = self.count + 1
        return self.count

def contains3(number):
    if number == 0:
        return False
    else:
        if number % 10 == 3:
            return True
        else:
            return contains3(number // 10)

def multipleOf7(number):
    return number % 7 == 0

class ZapBuzz(Counter):
    def __init__(self):
        Counter.__init__(self,0)
    def increment(self):
        result = Counter.increment()
        c3 = contains3(result)
        m7 = multipleOf7(result)
        if c3 and m7:
            return "zap␣buzz"
        if c3:
            return "buzz"
        if m7:
            return "zap"
        return result
```

Problem 5

```
class Taxi:
    # some additional methods for Taxi
    ...
    def distanceFrom(self,x,y): # or really rather in class Car
        dx = x - self.getLocationX()
        dy = y - self.getLocationY()
        d = (dx*dx + dy*dy)**0,5
        return d
    def canPickup(self,x,y):
        if self.getStatus():
            return False
        d = self.distanceFrom(x,y)
        if d / self.mpg > self.getGas():
            return False
        return True

class Dispatcher:
    def __init__(self):
        self.fleet = []
    def hire(self,taxi):
        self.fleet.append(taxi)
    def hail(self,x,y):
        closest = None
        distance = 0.0
        for taxi in self.fleet:
            taxi_distance = taxi.distanceFrom(x,y)
            if taxi.canPickup(x,y) and
                 ((closest is None) or taxi_distance < distance):
                 distance = taxi_distance
                 closest = taxi
        if closest is not None:
            closest.driveTo(x,y)
            closest.pickup()
        return closest
```

Problem 6

```
class LinkedList:
    ...
    def swap_at(self,position):
        follow = None
        current = self.first # Will be the first of the two
        while position > 1:
            current = current.next
            position -= 1
        next = current.next # Will be the second of the two
        #
        current.next = next.next
        next.next = current
        #
        if follow is None:
            self.first = next
        else:
            follow.next = next
```