ECS 10
Concepts of Computation
More Example Final Problems, Programming Solutions

Check your answers by running the programs.

1. Here is a little program, not necessarily correct. What does it print?

```
def capitalize(s):
     new = ""
     for c in s:
          if c in lowerCase:
               i = lowerCase.find(c)
               new = new+upperCase[i]
          else:
               new = new+c
     return new

lowerCase = "abcdefghijklmnopqrstuvwxyz"
upperCase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
myStr = capitalize("Happy Birthday")
print myStr
```

   a) `Happy Birthday`
   b) `HAPPY BIRTHDAY`
   c) `new`
   d) Nothing; the function crashes because `lowercase` is not local.

2. What could fill in the blank line?

```
 def maximum(a,b):
      if a > b:
           max = a
      else:
           max = b
      -----------------

myYOB = 1958
herYOB = 1983
if maximum(myYOB, herYOB) == her:
     print "She is younger"
```

   a) `return True`
   b) `return False`
   c) `return max`
   d) It can remain blank.

3. This program is supposed to print out the month and year for the next 30 months. What could fill in the blank?

```
def nextMonth(m):
     ----------------
     m = m+1
     if m>11:
          m = 0
          year = year+1
     return m

month = 11
year = 2009
for i in range(30):
     print month+1,year
     month = nextMonth(month)
```

a) `year = 2009`
b) `year = year`
c) `global year`
d) It could remain blank.

4. This program

```
def countLetters(s):
     for char in s:
          if char in D:
               D[char] = D[char]+1
          else:
               D[char] = 1

myStr="omle was I ere I saw elmo"
D = {}
countLetters(myStr)
for char in D:
     print char,D[char]
```

a) prints 10 lines
b) prints one line
c) crashes because char is both a local and a global variable.

5. Programming problem:

   Here is a program with the block inside a function definition missing. Write the complete definition of the function `lengths(L)` below. The function `lengths(L)` takes a list of strings `L` as its input argument, and returns a list containing the lengths of all the strings.

```
def lengths(L):
    --------------------
    --------------------
    --------------------

people = ["Anne","Katherine","James","Lu"]
print lengths(people)
```

   So this program should print

```
[4,9,5,2]
```

   Your function should work with any input list of strings, so that for instance if the program is changed by changing the list of strings to

```
people = ["Joe","Jose"]
```

   the output (without changing anything else in the function) should become

```
[3,4]
```

   Remember that if `x` is a string, the built-in function `len(x)` returns the number of characters in `x`, and if `x` is a list, `len(x)` returns the number of elements in `x`.

```
# Solution

def lengths(L):
    lenList = []  # list of lengths
    for s in L:    # s will be each string in L in turn
        lenList = lenList + [len(s)]  # concatenate new length onto list
    return lenList  # return list of lengths to main program

people = ["Anne","Katherine","James","Lu"]
print lengths(people)
```

6. Programming problem:

   Your friend in animal science has been making observations on how many pounds of food the four pigs in the pig barn eat each day, over the course of a week. He has kept his observations for each day in file called `pigs.csv`, which is eight lines long, and looks like this (the middle four lines are not shown):

```
Date,Porky,Heather,Rose,Gus,Bob
12/1,5.5,4.8,5.3,5.2,4.6
12/2,5.8,5.3,4.9,5.2,4.7
...
12/7,6.2,5.5,5.5,3.6,4.8
```

   You offer to write a program for him that will read in the data and respond to queries about how much a particular pig ate over the course of the week. For instance, running your program might look like this:

```
Enter name of pig: Heather
Heather ate 4.8, 5.3, ..., 5.5
Enter name of pig: Bob
Bob ate 4.6, 4.7, ..., 4.8
Enter name of pig: Petunia
There is no pig named Petunia
Enter name of pig:
Press enter to exit
```

   (Again, the middle four numbers in each response of the computer are not shown). Your program should use a dictionary.

```
# Tough problem; many solutions; here is one.

inFile = open("pigs.csv","r")
D = {}

for line in inFile:
    line = line.strip()
    items = line.split(",")

    if items[0] == "Date":  # The first line contains the names of the pigs
        pigList = items       # Save it so we can interpret following lines
        for i in range(1,len(items)):
            pig = pigList[i]
            D[pig] = []           # Dictionary gets empty list for each pig
    else:                          # Lines for each day
        for i in range(1,len(items)):   # amounts of food
            pig = pigList[i]                  # pig from saved first line
            D[pig] = D[pig]+[items[i]]  # add todays food to the list

while True:
    pig = raw_input("Enter name of pig: ")
    if pig == "":
        break
    if pig in D:
        print pig,"ate",
        ate = D[pig]
        for amount in ate:
            print amount,
        print "\n"
    else:
        print "There is no pig named",pig
```

7. Programming Problem:

   We will be given a list of prices, written as strings such as `"$12,350.34"` or `"$85.99"`. Write a function `dollars(s)` which takes such a string as its input argument, and returns an integer number of dollars, rounded down to the nearest dollar. So if you fill in your function in this program:

```
def dollars(s):
     ____________________
     ____________________
     ____________________
     ____________________
     ____________________

prices = ["$12,350.34","$85.99","$5,302.69"]
for i in range(0,3):
     print dollars(prices[i])
```

   it should print

```
12350
85
5302
```

```
# Filled in function
def dollars(s):
    s = s.replace("$","")
    s = s.replace(",","")
    s = s.replace(".","")  # get price in cents
    number = int(s)
    number = number/100  # then do integer division to get dollars
    return number

prices = ["$12,350.34","$85.99","$5,302.69"]
for i in range(0,3):
     print dollars(prices[i])
```