Spring 2010 Name: Key

Introduction to Computing Test 2

Question 1. (15 points) For the following strings, predict the results:

```
cheer = 'GO Panthers!!!'
rhyme = 'The cow jumped over the moon.'
        012345678911111111122222222
                  012345678901234567
```

Expression	Predicted Result
`cheer[0]`	'G'
`rhyme[1:6]`	'he co'
`cheer[4:-2]`	'anthers!'
`'jump' in rhyme`	True
`len(cheer)`	14
`cheer[0]*5.center(15)`	' GGGGG '
`rhyme.split()`	['The', 'cow', 'jumped', 'over', 'the', 'moon.']
`cheer.lower().strip('!')`	'go panthers'
`rhyme.isalpha()`	False

Question 2. (15 points) Write a program that prompts the user for two text-file names, and then reads the contents from the first file name into the second file name. (The original contents of the second file can be erased, but you should not need to do anything special to achieve this result.)

```
inFileName = raw_input("Enter file to copy: ")
outFileName = raw_input("Enter file to receive copy: ")
inFile = open(inFileName, 'r')
outFile = open(outFileName, 'w')
for line in inFile:
    outFile.write(line)
inFile.close()
outFile.close()
```

Question 3. Bank UNI has a menu-driven, teller program that maintains a text file `accountData.txt` of bank-account records. Each bank-account record is on a single line with 4 fields separated by commas (','). The order of the fields on a line is: a unique account number (a string of 6 digits), current balance, first name, and last name. For example, the start of the `accountData.txt` file might look like:

```
123456,234.87,John,Smith
345678,1087.12,Sue,Doe
        .
        .
        .
```

At the start of the day, the file "accountData.txt" is read into a list, accountsList, with each item in the list containing the information about an account as a list. For the above file, the `accountsList` would look like:

```
[['123456','234.87','John','Smith'], ['345678','1087.12','Sue','Doe'],...]
```

The main function for this teller program is:

```
def main():
    accountsList = readAccountFile('accountData.txt')
    mainMenu(accountsList)
    writeAccountFile('updatedAccountData.txt', accountsList)
```

a) (15 points) Complete the following `readAccountFile` function that takes a string for the file name and returns the `accountsList`.

```
def readAccountFile(fileName):
    acctFile = open(fileName,'r')
    accountsList = []
    for line in acctFile:
        acctList = line.strip('\n').split(',')
        accountsList.append(acctList)
    return accountsList
```

b) (15 points) Assuming `accountsList` has been read in from the file successfully, complete the function definition to linear search `accountsList` for a specified account number and returns the matching account list record (or None on an unsuccessful search). (The `targetAccountNumber` is given as a string of 6 digits.)

```
def searchByAccountNumber(accountsList, targetAccountNumber):
    for account in accountsList:
        if account[0] == targetAccountNumber:
            return account

    return None
```

Question 4. (15 points) The factorial function informally is n! = 1 x 2 x 3 x ... x n. For example, 5! = 1 x 2 x 3 x 4 x 5 = 120. A recursive definition of the factorial function is:

n! = n x (n-1)! for n ≥ 1, and
n! = 1 for n = 0.

Complete the following recursive function for factorial(n).

```
def factorial(n):
    if n == 0:
        return 1
    else
        return n * factorial(n-1)
```

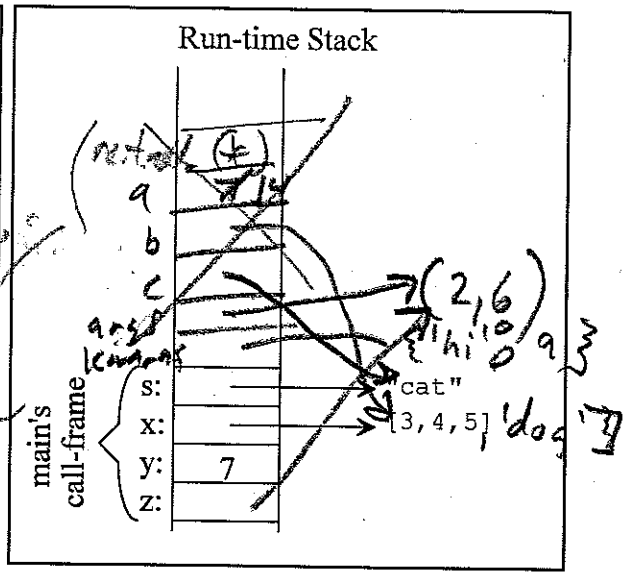
Name: _____

Question 5. (15 points) Complete the tracing of the following code to show the expected output and the run-time stack as the program executes. Recall that a function's call-frame contains the return address, formal parameter(s), and local variable(s).

```
# program to trace
def main():
      s = "cat"
      x = [3,4,5]
      y = 7
      z = doSomething(y,x,s,2,6,'hi'=9)
      print 's=',s,'x=',x,'y=',y,'z=',z

def doSomething(a, b, c = 10,*args,**kwargs):
      print 'a=',a,'b=',b,'c=',c,
      print 'args=',args,'kwargs=',kwargs
      a = a * 2
      b.append('dog')
      return args
main()
```

Continue trace from here



Expected Output

a=7 b=[3,4,5] c=cat args=(2,6) kwargs={'hi':9}
s=cat x=[3,4,5,'dog'] y=7 z=(2,6)

Question 6. (10 points) Predict the output of the following higher-order functions. Assume `mult` is definited as:

```
def mult(x, y):
    return x * y
```

a) `print reduce(mult, range(1,5))` [1,2,3,4]

24

b) `print map(len, ['test', 'number', '2'])`

[4,6,1]

c) `print filter(lambda s: s.isalpha(), ['test', 'number', '2'])`

['test', 'number']

d) Rewrite the "`reduce(mult, range(1,5))`" from part (a) using a lambda expression instead of the function `mult`.

print reduce(lambda x,y: x*y, range(1,5))