CSCI 150
Exam 2 Solutions
November 20, 2015

1. Here is a portion of a program that is supposed to build up a dictionary identifying people's favorite colors. Since a person might have more than one favorite, the dictionary associates a list of colors with each person it knows.

```
def AddToDictionary(D, person, color):
    if person in D.keys():
        D[person].append(color)
    else:
        D[person] = color

def main():
    FavoriteColors = {}
    AddToDictionary(FavoriteColors, "bob", "purple")
    AddToDictionary(FavoriteColors, "Mary", "blue")
    AddToDictionary(FavoriteColors, "Frodo", "green")
    AddToDictionary(FavoriteColors, "Mary", "pink")
    AddToDictionary(FavoriteColors, "Frodo", "red")

main()
```

When I run this I get an error message pointing to the line D[person].append(color), which I have printed in bold, saying that a 'str' object has no atribute 'append'.

a) Explain in one sentence what is wrong. **The first time a person is added to the dictionary, the value associated with that person is a string, not a list. The next time you get a color for that person it tries to append onto this string, with the resulting error message.**
b) Indicate in the code how you would change the program to fix it. **The else clause in function AddToDictionary( ) should say D[person] = [color]**

2. Here is part of the definition of a class that represents fractions:

```
class Fraction:
    def __init__(self, numerator, denominator):
        self.value = numerator/ denominator

    def Print(self):
        print( "%d/ %d = %.2f"%(numerator, denominator, self.value))

def main():
    x = Fraction(3, 4)
    x.Print()

main()
```

When I run this program I get an error message on the print statement in the class's Print method:

```
print( "%d/%d = %.2f"%(numerator, denominator, self.value))
NameError: name 'numerator' is not defined
```

Explain this error. You don't need to say how to fix it but you should explain why it finds variable numerator to be undefined. Isn't it defined in the __init__ method?

The variables numerator and denominator in the __init__( ) method cannot be seen in the Print( ) method; only instance variables are shared by all of the methods. If you wanted to fix the bug, in __init__( ) save numerator and denominator as self.num and self.denom, then change Print( ) to refer to (self.num, self.denom, self.value)

3. What will this program print?

```
def foo(n):
        if n == 0:
                return 0
        elif n%2 == 0:
                return 1+foo(n//2)
        elif n%3 == 0:
                return 2+foo(n//3)
        else::
                return foo(n-1)

def main():
        print( foo(21) )
main()
```

It prints 5:

foo(21) = 2+foo(7)
= 2+foo(6)
= 2+1+foo(3)
=2+1+2+foo(1)
= 2+1+2+foo(0)
= 2+1+2+0
= 5

4. Write a program that opens file “F.txt” and prints how many times each letter a through z appears in the file. You did something like this in Lab 07 for the Decrypt.py program. For this you should consider capital letters and their lower-case versions to be the same letter, so the count for ‘a’ includes both instances of ‘a’ and of ‘A’.

```
def main():
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    D = {} # keys are letters, values are counts
    for x in alphabet:
        D[x] = 0
    F = open( "File.txt", "r")
    for line in F:
        for letter in line.lower():
                if letter in alphabet:
                D[letter] = D[letter] + 1
    for x in alphabet:
        print( x, D[x] )

main()
```

5. Write a recursive function RemoveSpaces(s) that takes as an argument a string s and returns a string like s with all of its spaces removed. For example, RemoveSpaces(“President Marvin Krislov”) returns “PresidentMarvinKrislov”.

```
def RemoveSpaces(s):
    if s == "":
        return s
    elif s[0] == " ":
        return RemoveSpaces( s[1:] )
    else:
        return s[0] + RemoveSpaces( s[1:] )
```

6. Suppose we have a class Album that represents jazz recordings. The Album class has instance variables self.leader, self.albumName and self.date, where *leader* is a band leader such as "Oliver Nelson", *albumName* is a string such as "The Blues and the Abstract Truth", and *date* is the year of recording, such as 1961. Let's assume that class Album has a __str__(self) method that formats this information in a nice way. **You don't need to write class Album; just assume it exits.**

   You do need to write class Collection, which describes a collection of albums.
   The constructor for Collection is

   __init__(self, owner_name)

   where owner_name is the person who owns the collection, such as "bob". To add an album to the collection we use

   addAlbum(self, a)

   where a is an object of class Album. Give class Collection a **Print(self)** method that prints all of the albums in the collection. Also give class Collection a **SearchByDate(self, year)** that prints all of the albums in the collection that were recorded in the given year.

```
class Collection:
    def __init__(self, owner_name):
        self.owner = owner_name
        self.albums = [ ]

    def addAlbum(self, a):
        self.albums.append(a)

    def Print(self):
        for album in self.albums:
            print(album)

    def SearchByDate(self, year):
        for album in self.albums:
            if album.date == year:
                print(album)
```