CSCI 150
Exam 2
November 20, 2017

1. What will this print?

```
def A(n)
    print(n)
    if n > 1:
        if n%2 == 0:
            A(n//2)
        else:
            A(n+1)

def main():
    A(13)
main(  )
```

13
14
7
8
4
2
1

2. Here is a program that tries to copy a list:

```
def copy(L):
    M = []
    for i in range(0, len(L)):
        M[i] = L[i]
    return M

def main():
    L = copy( [1,2,3] )
    print(L)

main()
```

When I try to run this it gets an error message:

```
M[i] = L[i]
IndexError: list assignment index out of range.
```

a) Explain what this error message means

On the first time this tries to copy a number from L to M, list M is empty, so there is no M[i].

b) How should the code be changed so that it makes a copy of the list [1,2,3] and prints [1,2,3]?

Change the line M[i]=L[i] to M.append(L[i])

3. Here is a program that doesn't work. This program tries to make a dictionary that associates to each person a list of their phone numbers:

```
def add(D, name, phone):
    # The dictionary D is structured so its keys are names
    # and its values are lists of phone numbers. This appends
    # the given phone number onto the list D[name].

    if name in D.keys():
        D[name].append( phone )
    else:
        D[name] = phone

def main():
    Phones = {}
    add( Phones, "bob", "774-1977" )
    add( Phones, "Rumplestiltskin", "000-0000" )
    add( Phones, "bob", "775-8386" )
    add( Phones, "Harry", "I'm magical and don't need a phone" )

main()
```

When I run this program I get the error message

```
D[name].append( phone )
AttributeError: 'str' object has no attribute 'append'
```

a) Explain what this error message means

This adds a new name to the dictionary with the line D[name] = phone, where phone is a string. That makes the value associated with the name a string. The next time we add a phone to this name it tries to append the new phone number onto D[name], but you can't append onto a string.

b) How should the code be changed so it stores all of the phone numbers? Either write the new code here or cross out and write over the code above.

Change the line D[name] = phone to D[name] == [phone] to make the value associated with the name be a list of phone numbers.

4. Here is a program that tries to make a list of pets.

```
class Pet:
        def __init__(self, name, species):
                self.name = name
                self.species = species

        def Print(self):
                print( "%s is a %s"%(name, species) )

def main():
                L = [ ]
                L.append( Pet( "Jinx", "Cat" ) )
                L.append( Pet( "Rover", "Dog" ) )
                for x in L:
                        x.Print( )

main()
```

Something is wrong with the class definition because when I run this program I get the error message

```
print( "%s is a %s"%(name, species) )
NameError: name 'name' is not defined
```

How should this program be fixed? Again, either rewrite the code here or change it above.

Change the print statement to print("%s is a %s"%(self.name, self.species))

5. Write a program that will read one string and print out the most frequent letter in that string. If I give it the string "abbacabbadb" it should say "The most common letter is 'b'." If there is a tie for the most common frequency your program could print any of the winners. You can assume the string contains only lower-case letters.

```
def main( ):
        s = input( "string: " )
        Counts = { }
        for letter in s:
                if letter in Counts.keys( ):
                        Counts[letter] = Counts[letter] + 1
                else:
                        Counts[letter] = 1
        bigLetter = s[0]
        for letter in Counts.keys( ):
                if Counts[letter]  > Counts[bigLetter]:
                        bigLetter = letter
        print( "The most common letter is %s." % bigLetter )
main( )
```

Of course, this can be broken into several functions: one to build the Counts dictionary and one to find the key with the largest value. Note that it isn't necessary to sort the letter counts, though an alternative way to do this is to put the numerical counts into a list, sort the list, and then find a key whose value is the last (largest) value of this sorted list.

6. Write a recursive function pal(s) that returns a palindrome made out of string s by putting each letter of s except the last at the front and back of the result. So pal(“abc”) returns “abcba”, pal(“a”) returns “a” and pal(“abcde”) returns “abcdedcba”. To get full credit function pal( ) needs to be recursive. You will get some credit for doing this with a loop, but it is probably easier to do recursively.

```
def pal(s):
    if len(s) <= 1:
        return s
    else:
        return s[0] + pal( s[1:] ) + s[0]
```