CSCI 150

Exam 1 Solutions

1. What does this print? Read it carefully.

```
def increase(x):
    x = x+10
    return x

def main():
    x = 5
    if x < 10:
        increase(x)
    print(x)

main()
```

This prints 5. The function increase(x) has no way to change the variable x in main( ); the only assignment to x in main( ) is x=5/

2. The following program tries to find and print the next prime number after 25, which is 29. Unfortunately, it crashes with an error message that points at the line print(nextP). The error message says that nextP is not defined. Doesn't the line nextP = n within functionn nextPrime( ) define nextP? Explain this error message in one or two sentences. You don't need to fix the program; just explain why nextP is not defined.

```
def isPrime(x):
    # This tries to say if x is a prime number
    for d in range(2,x):
        if x%d == 0:
            return False
     return True

def nextPrime(n):
    # This looks for the next prime number after n
    x = n+1
    while not isPrime(x):
        x = x+1
    nextP = x



def main():
    nextPrime(25)
    print( nextP )

main()
```

Function nextPrime(n) has a variable nextP, but main( ) can't see it because functions can't see the variables of any other functions. There is no variable nextP in main( ).

3. Consider the following program, which has a recursive function count( ):

```
def printer(n):
    if n > 0:
        print(n)

def count(n):
    if n == 0:
        print( "I'm done!" )
    else:
        printer(n)
        count(n-2)

def main():
    count(6)

main()
```

If I call count(6) in main( ) the program prints

```
6
4
2
I'm done!
```

However, if I call count(7) I get pages and pages of error messages, then the program crashes, finally saying “maximum recursion depth exceeded”. Explain in one or two sentences what is wrong. You don’t need to correct the program, just explain why it doesn’t work.

Recursive functions have to ensure that their arguments eventually reach the base case. With function count(n) that happens with even numbers – if you start with any even number and subtract 2 from it enough times you will eventually get to 0. That doesn’t happen with odd numbers – you eventually get to 3, then to 1, then to 1 then to -1, and after that to more negative numbers. With odd arguments you never get to the base case, 0.

4. Here is a program that is supposed to check for palindromes.

```
def reverse(s):
    rev = ""
    for letter in s:
        rev = letter + rev
    print( "The reversal of %s is %s."%(s, rev) )
    return rev

def isPalindrome(str):
    if str == reverse(str):
        print( "%s IS a palindrome"%str )
        return True
    else:
        print( "Nope. %s IS NOT a palindrome."%str )
        return False

def main():
    if isPalindrome("bob"):
       print( "Yippee" )

main()
```

When I run this is prints

```
The reversal of bob is bob.
Nope. bob IS NOT a palindrome
```

a) Explain in one sentence why this says "bob" is not a palindrome when it has already said "bob" is the reversal of "bob".

The function reverse(s) doesn't return anything, so the line
if str == reverse(str)
in main( ) becomes
if str == None
which is False.

b) Fix the program so it says "bob" is a palindrome and then prints "Yippee" in main( ). You don't need to rewrite the program; cross out lines or add code to the text above.

See the lines in bold in the program. We need to add return statements to both reverse( ) and isPalindrome( ).

5. Write a program that repeatedly asks the user for numbers; the input ends when the user enters 0. The program then prints the sum of the numbers it was given. Here is a sample run, where the text in bold is printed by the computer:

   number: 3
   number: 14
   number: 25
   number: 0
   Those sum to 42

```
def main( ):
    sum = 0
    done = False
    while not done:
        n = eval(input("Number? "))
        if n == 0:
            done = True
        else:
            sum = sum + n
    print( "The sum is %d."%sum )

main()
```

6. Write a recursive function noSpace(s) that takes a string argument s and returns a new string just like s but with the spaces removed. So noSpace(“bob”) returns “bob”, noSpace(“Marvin Krislov”) returns “MarvinKrislov” and noSpace(“a b c d e f g”) returns “abcdefg”.

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

7. Write a function moreBobs(s, t) that takes two string arguments, s and t and returns the one with more instances of "bob". If neither has any instances of "bob" the function should return "bobless". If they have the same number of instances of "bob" it should return "tie". For example, moreBobs("bob is the bob", "bobobob") should return "bobobob", (successive instances of "bob" can share letters so "bob is the bob" has 2 instances while "bobobob" has 3) and moreBobs("bob is the blob", "shishkabob") should return "tie".

```
def countBobs(s):
    # This returns the number of "bob" instances
    # in string s
    count = 0
    for i in range(0, len(s)-2):
        if s[i:i+3] == "bob":
            count = count + 1
    return count

def moreBobs(s, t):
    ns = countBobs(s)
    nt = countBobs(t)
    if ns == 0 and nt == 0:
        return "bobless"
    elif ns == nt:
        return "tie"
    elif ns > nt:
        return s
    else:
        return t
```