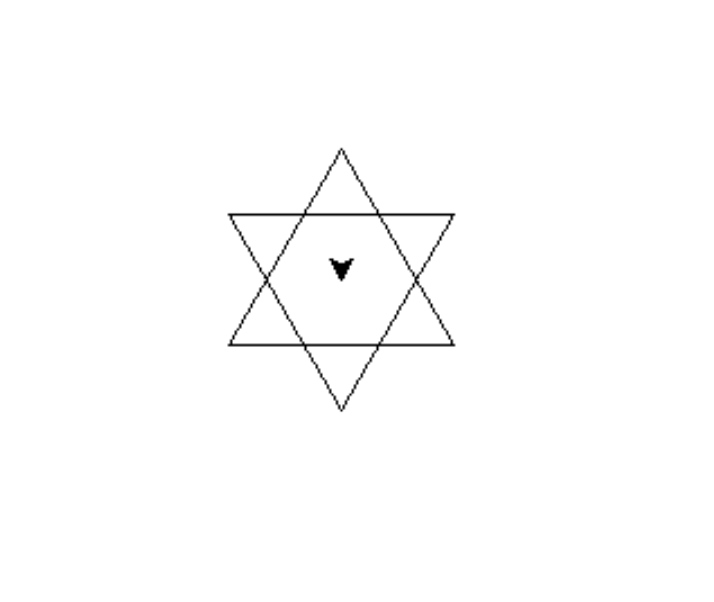
CSCI141 -07 PRACTICE EXAM

1. Draw to the right of the code, what the turtle would draw when executing main below. Assume proper imports have been done, and that the turtle starts at the center; pen down, facing east.

import turtle as tt

LEN1 = 100

LEN2 = LEN1 / (2 \* math.sqrt(3) ) # about 29



def draw\_part( ):

tt.forward( LEN2 )

tt.pendown( )

tt.right(90)

tt.forward(LEN1 / 2)

tt.right(120)

tt.forward(LEN1)

tt.right(120)

tt.forward(LEN1)

tt.right(120)

tt.forward(LEN1 / 2)

tt. left(90)

tt.penup( )

tt.backward( LEN2)

def main( ):

tt.penup( )

tt.left( 90)

draw\_part( )

tt.left(180)

draw\_part( )

Is the turtle state changed from the start to the end of **draw\_part**? If so, how?

Doesn’t change

Is the turtle state changed from the start to the end of **main**? If so, how?4

Turtle ends 90 degrees to the right of original state

1. For each expression, write the value it would evaluate to, the value’s type (str, int, float, bool). If the expression is invalid, write ERROR for the resulting value.

| h = ‘hello’ | w = ‘world’ | z = 0 | x = 15 | y = 37 |
| --- | --- | --- | --- | --- |
| n = 2 | t = ‘22’ | d = ‘d’ | c = ‘CompSci144EIOU’ |  |

| **Expression** | **Resulting value or ‘ERROR’** | **Type of value or ‘ERROR’** |
| --- | --- | --- |
| h + w | helloworld | string |
| x / 2 | 7.5 | float |
| x // 2 | 7 | int |
| x / z | ERROR |  |
| n \* t | ‘2222’ | str |
| n \* x | 30 | int |
| w + y | ERROR |  |
| int( t ) - y | -15 | int |
| h[ 1 ] | ‘e’ | str |
| h[ 1 : ] | ‘ello’ | str |
| h[ : 1] | ‘h’ | str |
| c[ 0 : 13 : 4 ] | ‘CS4O’ | str |
| c[7 : 12] | ‘144EI’ | str |
| x == 15 | True | bool |
| z > x | False | bool |
| d < ‘j’ | True | bool |
| t > x | ERROR |  |
| True or False | True | bool |
| True and False | False | bool |
| len( h + ‘,’ + w ) | 11 | int |

1. Compute the results of calling the function below

def rek( n ):

if n == 0:

return 1

else:

t = lupe( n )

return t \* rek( n - 1 )

def lupe( n ):

s = 0

while n > 0:

s = s + n

n = n - 1

return s

| **n** | **result of calling lupe( n )** | **result of calling rek( n )** |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| 2 | 3 | 3 |
| 3 | 6 | 18 |
| 4 | 10 | 180 |

[ TRUE / FALSE ] rek is **tail** recursive

False

Rewrite lupe using a **range based for loop**

def lupe\_for\_loop( n ):

total = 0

for i in range(n + 1):

total += n

return total

1. Given the tail recursive function reverse and it’s helper function **reverse\_rec**,

def reverse ( s1 ):

return reverse\_rec(s1, “”)

def reverse\_rec(s1, acc):

if s1 = “”

return acc

else:

return reverse\_rec(s1[ 1 : ], s1[ 0 ] + acc)

finish the substitution trace below.

reverse( “ABD” ) = reverse\_rec( “ABD”, “”)

= reverse\_rec( “BD”, “A”)

= reverse\_rec( “D”, “BA”)

= reverse\_rec( “”, “DBA”)

= “DBA”

The above algorithm is tail recursive. Rewrite it as a single function using a **while loop**.

def reverse\_rewrite2(str):

reverse = ""

count = len(str) - 1

while count >= 0:

reverse += str[count]

count -= 1

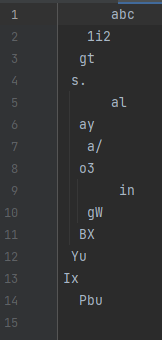
return reverse

# return str[::-1] would not be acceptable for this question because it specifically asks for a while loop

5. There is a file, named **info.txt**. Get every second letter in every line of the file and print it out in one line.

*Hint:* you can use *end=””* as the second parameter for print to get everything in one line

Here’s what **info.txt** looks like



def read\_info():

fd = open(‘info.txt’, ‘r’)

for line in fd:

separated = line.strip()

print(separated[1], end = “”)