**HOMEWORK 5 – Recursion / Exceptions**

Answer the following questions. **THIS ASSIGNMENT WILL BE DUE AT A DATE TO BE ANNOUNCED IN CLASS.**  (points as noted - 100 points total)

1. What is the output of the following code? (10 points)

def function (num):

if (num > 0):

for x in range(0,num):

print('\*',end=' ')

print("\n")

function(num-1)

def main ():

x = 10

function(x)

main()

**The Output for this program is:**

\* \* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

2. Ackermann’s function is a recursive mathematical algorithm that can be used to test how well a computer preforms recursion. Write a function A(m,n) which solves Ackermann’s Function. Use the following logic in your function:

* If m = 0 then return n+1
* If n = 0 then return A(m-1, 1)
* Otherwise, return A(m-1, A(m, n-1))

Test your function in a program that displays the following values: A(0, 0), A(0,1), A(1,1), A(1,2), A(1,4), A(2,2), A(3,2). (10 points)

**The function A(m, n) would be as follows:**

def A(m, n):

if m == 0:

return n+1

elif n == 0:

return A(m-1, 1)

else:

return A(m-1, A(m, n-1))

if \_\_name\_\_ == "\_\_main\_\_":

print("A(0, 0) =", A(0, 0))

print("A(0, 1) =", A(0, 1))

print("A(1, 1) =", A(1, 1))

print("A(1, 2) =", A(1, 2))

print("A(1, 4) =", A(1, 4))

print("A(2, 2) =", A(2, 2))

print("A(3, 2) =", A(3, 2))

3. Write a recursive function to compute the nth term of the sequence defined by the recursive relation an = an-1 + an-2 + an-3, where a0 = 1, a1 = 1, and a2 = 1, and n = 3, 4, 5, ... . Then, using the approach that we used in Class 15, identify what happens to the ratio an/an-1 as n gets larger. Does there appear to be a “golden ratio” for this recursively defined function? (30 points)

The recursive function would be:

def find\_nth\_term(n):

if n == 0:

return 1

elif n == 1:

return 1

elif n == 2:

return 1

else:

return find\_nth\_term(n-1) + find\_nth\_term(n-2) + find\_nth\_term(n-3)

if \_\_name\_\_ == "\_\_main\_\_":

print(find\_nth\_term())

As “n” gets larger in the ratio of an / an-1 using the function find\_nth\_term() the output converges to 1.8392…… This suggests a golden ratio for the recursively defined function.4. What is the output of the following code, assuming that the following values are inputted: 0, 8, ‘aad43,’ and -3. (10 points)

numerator = 5

try:

denominator = float(input("Please enter your denominator: "))

if(denominator<0): throw

value = ((5/denominator)\*\*0.5)

print("The square root of 5.0/",denominator," is ",sep='',end='')

print(format(value,'.4f'),".",sep='')

except IOError:

print("Error opening file!")

except ValueError:

print("Non-numerical value entered!")

except ZeroDivisionError:

print("Cannot divide by zero!!!")

except:

print("Other error!")

else:

print("It worked! ",end='')

finally:

print("Done.")

Running the program with those inputs would give the outputs of:

**For 0:**

Cannot divide by zero!!!

Done.

**For 8:**

The square root of 5.0/8.0 is 0.7906.

It worked! Done.

**For ‘aad43’:**

Non-numerical value entered!

Done.

**For -3:**

Other error!

Done.

5. Write an exception handler to handle the natural logarithm function. Your code should prompt the user to enter a positive value, then have the exception handler take care of the case where the argument is not positive. Have the program output the natural logarithm of the input value with 4 decimal places displayed. Prompt the user to enter additional values if the user so desires. (40 points)

**The program would be:**

import math

while True:

choice = input("Do you want to enter a positive value? (y/n): ")

if choice.lower() == 'n':

break

try:

x = float(input("Enter a positive value: "))

if x <= 0:

raise ValueError("Value must be positive")

else:

result = math.log(x)

print("ln({:.4f}) = {:.4f}".format(x, result))

except ValueError as ve:

print("Error: {}".format(ve))

except Exception as e:

print("An error occurred: {}".format(e))