NORTH AMERICAN
UNIVERSITY
INSPIRATION INNOVATION GLOBAL COMPETENCE

COMP 3317
Algorithms

Quiz #1

Name: Geraldo Braho

Please implement the following pseudo codes in your own programming language. Paste your code below each question.

1. Getting Fairness from Biased Sources:

```
Loop for 10 times
Flip the biased coin twice.
If the result is {Heads, Tails}, return Heads.
If the result is {Tails, Heads}, return Tails.
If the result is something else, start over.
```

Hint: Biased coin can be implemented as follows in Python.

```
def BiasedCoin():
   number = random.randint(1,4)
   if number == 3:
       return "tail"
   else:
       return "head"

def flipCoin(BiasedCoin):
     flip1, flip2 = BiasedCoin(), BiasedCoin()
     if flip1 == "head" and flip2 == "tail":
        return "head"
     elif flip1 == "tail" and flip2 == "head":
        return "tail"
     else:
        return flipCoin(BiasedCoin)


result= []
for i in range(10):
   i = flipCoin(BiasedCoin)
   result.append(i)


print flipCoin(BiasedCoin)
print result
```

2. Randomize an Array:

```
RandomizeArray(String: array[])
```

```
Integer: max_i = <Upper bound of array>
For i = 0 To max_i - 1
// Pick the item for position i in the array.
Integer: j = <pseudorandom number between i and max_i inclusive>
<Swap the values of array[i] and array[j]>
Next i
End RandomizeArray
```

```
def RandomizeArray(listN):
    max_i = list[len(listN) - 1]
    print(listN)
    print(max_i)
    print(len(listN))
    i = 0
    for i in range(len(listN)):
        j = random.randint(i,(len(listN) - 1))
        temp = listN[i]
        listN[i] = list[j]
        listN[j] = temp
    return list
```

3. Calculate A to the Power P:

```
// Calculate A to the power P.
Float: RaiseToPower(Float: A, Integer: P)
<Use the first fact to quickly calculate A, $A^2$, $A^4$, $A^8$, and so on until you get to a value $A^N$ where N + 1 > P>
<Use those powers of A and the second fact to calculate $A^P$>
Return $A^P$
End RaiseToPower
```

```
import random
def calc_power(b,p):
       val = b
       for i in range (b,p+1):
          val = val*b
       return val
b=(random.randint(1,10000))
p=(random.randint(1,10000))
numP=calc_power(b,p)
print b,p
print numP
```

4. Finding Prime factors:

```
List Of Integer: FindFactors(Integer: number)
List Of Integer: factors
// Pull out factors of 2.
While (number Mod 2 == 0)
factors.Add(2)
number = number / 2
End While
// Look for odd factors.
Integer: i = 3
Integer: max_factor = Sqrt(number)
While (i <= max_factor)
// Pull out factors of i.
While (number Mod i == 0)
// i is a factor. Add it to the list.
factors.Add(i)
// Divide the number by i.
number = number / i
// Set a new upper bound.
max_factor = Sqrt(number)
End While
// Check the next possible odd factor.
i = i + 2
End While
// If there's anything left of the number, it is a factor, too.
If (number > 1) Then factors.Add(number)
Return factors
End FindFactors
```

```
def prime_factors(n):
    i = 2
    factors = []
    while i * i <= n:
        if n % i:
            i += 1
        else:
            n //= i
            factors.append(i)
    if n > 1:
        factors.append(n)
    return factors
```

5. The least common multiple (LCM) of integers A and B is the smallest integer that A and B both divide into evenly. How can you use the GCD to calculate the LCM?

GCD Pseudocode:

```
Integer: GCD(Integer: A, Integer: B)
While (B != 0)
Integer: remainder = A Mod B
// GCD(A, B) = GCD(B, remainder)
A = B
B = remainder
End While
Return A
End GCD
```

```
def gcd(a, b)
    while (b!= 0):
       reminder= b
       b = a% b
       a = reminder
    return a
```

```
#another way
def gcd(a, b)
    while (a!=b):
        if a > b
           a = a − b;
        else
           b = b − a;
    return a
```

```
#   another way
def gcd(a, b)
    if b = 0
       return a
    else
       return gcd(b, a % b)
```

Please paste LCD implementation here:

```
def GCD(a,b):

    while(b != 0):
        remainder = a % b
        a = b
        b = remainder
    return a
```

```
print "gcd " + str(GCD(500,245))

def LCM(a,b):
     return ((a * b) / GCD(a,b))
print "LCM "+ str( LCM(500,245))

‘’’
GCD(a,b) = |axb|/LCM(a,b)
LCM(a,b)=|axb|/GCD(a,b)
‘’’
```