COMP 6481 – Assignment 2

//----------------------------------------------

//

// Assignment 2

// Tavtej Singh Lehri

// Written by: Tavtej Singh Lehri 40121745

//----------------------------------------------

Part I:

Question1:

1. Algorithm FlippedPair(A,n)

Input Array A of size n, (n>=1)

M <- 3

For (int i=0; i<n; i++)

{

For (int j=i+1; j<n; j++)

{

If(A[j] = A[i] + m )

Print (“[“+ A[i] + “,” + A[j] + “]”)

}

}

1. Time complexity of O(n^2)
2. Yes, this can be performed with time complexity of O(nlogn) with sorting and binary search as follows:  
   Algorithm: flipped(A,n)

Input Array A of size n, (n>=1)

Sort(A,n)

m<-3

For (int i=0; i<n; i++)

{

K= A[i]

If (K>=0)

{

Int x= binarySearch(A, 0, n-1, K)

If (x = K + m)

{

Print (“[“+ A[i] + “,” + A[x] + “]”)

}

}

}

1. O(n) is possible

filpped(A,n)

Input Array A of size n, (n>=1)

Take Hash Table H of size n

For (i = 0 to n-1)

{

int x = 0 - A[i]

if(x>=0)

{

if (table.contains(x))

{

Print ("[" + arr[i] + "," +x + "]")

}

}

}

Question 2:

1. Algorithm findPrime(A,n)

Input Array A of size n

Output pairs of numbers whose sum is prime

if num<-0 or num<-1 do

return false

for i <- 2 to i\*I <- num

if num%i <- 0 do

return true

return false

1. Algorithm findPrime(A,n)

Input Array A of size n

Output pairs of numbers whose sum is prime

Int Count <- 0

For i <- 0 to n

For j <- i+1 to n

Int sum = A[i] + A[j];

If(isPrime[sum])

count++;

1. time complexity for iterative approach is O(sqrt(sum) \* n^2)

for recursive approach is O(log n)

1. Use recursion to get all permutations

Question 3:

Algorithm duplicateElementLeft(A, n)

Input Array A of size n

Output BT with duplicate values at left node

Node root <- null

for i <-1 to n do

root <- add(root, i)

add(root, i)

Node current <- root

Node parent <- null

if root <- null do

return new Node(i)

while current != null do

parent <- current

if i <= current.get(i) do

current <- current.left

else

current <- current.right

if i<= parent.get(k) do

new Node(i) <- parent.left

else

new Node(i) <- parent.right

return root

**The resulting tree will be BST if there are no duplicates.**

//Referenced from www.geeksforgeeks.com