**Problem 1**

1. f(n) = O(g(n))
2. f(n) = O(g(n))
3. --------
4. f(n) = Θ(g(n))
5. f(n) = Ω(g(n))
6. f(n) = O(g(n))
7. f(n) = Θ(g(n))
8. f(n) = O(g(n))
9. f(n) = Ω(g(n))
10. f(n) = O(g(n))

**Problem 2**

Part 1

Prove by induction

Assume for all n < k.

Base case: ,

For n > 0,

Part 2

Prove that

Part 3

--------

**Problem 3**

Part 1

Part 2

Part 3

--------

**Problem 4**

Part 1

// create a new function passing in an array of n distinct elements and return the closest pair of values

function closestPair ( array A )

sort array A in order of size from smallest to largest

// Set the initial value to the difference of the first 2 elements of the array

set diff = A[1] – A[0]

LOOP through the array elements

// Check to see if the difference for the next set of values is < diff

// if so, set diff to this value

IF A[i+1] – A[i] < diff

set diff to A[i+1] - A[i]

set x to a[1]

set y to a[i+1]

ENDIF

Increment i

ENDLOOP

return x and y

End function

Part 2

// create a new function given an array A find the max value and return it

function findMax ( array A )

// Initialize the maximum value to be the first element value

Set max to A[0]

LOOP through the array of elements

// Compare the next element value to the max value.

// If next element is larger then set max to this value

IF A[i] > max

set max to A[i]

ENDIF

Increment i

ENDLOOP

Return max

END function

Part 3

// create a new function given an array A determine if there is a set of triplets

// such that A[i] + A[j] = [k]

function findTriplets ( array A )

LOOP through the array of elements

set v1 to A[i]

set v2 to A[i+1]

set v3 to A[i+2]

IF v1 + v2 equals v3

// exit out of function and return the 3 values

return v1, v2, and v3

ENDIF

Increment i

ENDLOOP

return “no such triplet exists”

END function