**Check 156 of Combined Notes PDF**

int usrInp;

list A

list B

ask user **FOR** length of list

store into usrInp;

**FOR** n = 0, …, usrInp:

ask user **FOR** single int input **FOR** list A of element n+1

store int into list A

**FOR** n = 0, …, usrInp:

ask user **FOR** single int input **FOR** list B of element n+1

store int into list B

**FUNCTION** bubbleSort(: real numbers with n 2):

**FOR** i:= 1 **to** n-1

**FOR** j:= 1 to n-i

**If** a[j] > a[j+1]

temp1 = a[j]

temp2 = a[j+1]

a[j+1] = temp1

a[j] = temp2

***//OUTPUT IS IN THIS FUNCITON***

**FUNCTION** bubbleSortCheckDuplicates(: real numbers with n 2):

//Not sure if this will work. It does a bubble sort that will delete

**FOR** i:= 1 **to** n-1

**If** a[i] = a[i+1]

delete a[i]

**FOR** i:= 1 **to** n-1

Output a[i]

**FUNCTION** Complement(list A)

//Just say that the complement of A is everything but for x in range(a) where you output a[x].

**FUNCTION** AUnionB(list A, list B)

list temp

//Loops through list A and appends to temp

**FOR** i:=0 **to** max length of list A or B:

add A[n] to temp

//Loops through list B to append B into temp.

**FOR** i:=0 **to** max length of list A or B:

//Loops through temp

**FOR** j:=0 to max length of temp

//if B[i] is not founded in temp, add B[n] to temp. Or basically, since we already added all of A, we just don’t want to add another duplicate from B into temp that was already added from A.

**IF** B[i] != temp[j]

add B[n] to temp

BubbleSort (temp) //This function will use a bubble sort function to organize function

BubbleSortCheckDuplicates (temp) //This function will use a bubble sort to check for duplicates

//This functions checks to see if there is an element in A and B that matches each other. If so, it would need to be outputted. This will check element of A with the element of B. In this case, if we are given A = [1, 2, 3, 4] and B = [4, 3, 5, 6], A[0] would be compared with B[0], B[1], B[2] and B[3]. Then A[1] would be compared with B[0], B[1], B[2], and so on. If there are any matches, said value is to be outputted. Otherwise, I could also put the value into a temporary list which will be passed into a function that will implement a bubble sort to output the list in numerical order.

**FUNCTION** AIntersectionB(list A, list B)

list temp

//Cycles through list A

**FOR** i:=0 **to** max length of list A:

//Cycles through list B.

**FOR** j:=0 **to** max length of list A:

//If A[i] is founded in B

**IF** A[i] = B[j]:

**append** A[i] or B[j] **to** temp

BubbleSort (temp) //This function will use a bubble sort function to organize function

BubbleSortCheckDuplicates (temp) //This function will use a bubble sort to check for duplicates

//This functions checks to see if there is an element that **is not BOTH in A and B** but rather **ONLY IN A or B**. If so, it would need to be outputted. A nested loop is used to compare the elements in A with B. If there are any matches, said value is to be outputted or put into a temporary list which will be passed into a function that will implement a bubble sort to output the list in numerical order.

**FUNCTION** ADifferenceB(list A, list B)

list temp

//Cycles through list A

**FOR** i:=0 **to** max length of list A:

//Cycles through list B.

**FOR** j:=0 **to** max length of list B:

//If A[i] is not founded in B

**IF** A[i] != B[n]:

**append** A[i] to temp

BubbleSort (temp) //This function will use a bubble sort function to organize function

BubbleSortCheckDuplicates (temp) //This function will use a bubble sort to check for duplicates

**FUNCTION** ASymmetricDifferenceB(list A, list B)

ADifference(A, B)

ADifference(B, A)