**(PART I) Program**

def merge\_Sort(A, B):

#Merges the two arrays

#Sorts merged arrays

A.extend(B)

A.sort()

return A

def checkInput(userInput):

#Checks if input length > 0

#Removes spaces from user input

#Checks if user input is all digits

#If not, warn user and return false

#Checks if user input is sorted

#If not, warn user and return false, otherwise return true

if(len(userInput) is not 0):

checkDig = checkDigits(userInput)

if(checkDig):

if(userInput == sorted(userInput)):

return True

else:

print("Please enter sorted array!")

else:

print("Please enter an array!")

return False

def checkDigits(userInput):

#Loops through each element of input array

#Strips leading "-" to check if element is digit

#If digit, convert string to int

#Otherwise warn user and return false

for num in range(len(userInput)):

if(userInput[num].lstrip('-').isdigit()):

userInput[num] = int(userInput[num])

else:

print("Please enter numbers only!")

return False

return True

def main():

checkA = False

checkB = False

#Get user input

#Convert string input into list

#Check if input is valid

#Set checkA/checkB to True if input is valid and break out of while loop

while(not checkA):

userInput = raw\_input("Enter First Array (Leave Spaces Between Numbers)\n")

arrayA = userInput.split()

checkA = checkInput(arrayA);

while(not checkB):

userInput = raw\_input("Enter Second Array (Leave Spaces Between Numbers)\n")

arrayB = userInput.split()

checkB = checkInput(arrayB)

sortedArray = merge\_Sort(arrayA, arrayB)

print(sortedArray)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**(Part II) Oracle**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Case # | Description | Input(Array A) | Input(Array B) | Expected Output | Actual Output | Result  (Pass/Fail) |
| 1 | Expected I/O | 1 3 5 7 | 2 4 6 8 | [1,2,3,4,5,6,7,8] | [1,2,3,4,5,6,7,8] | Pass |
| 2 | Empty Array A | N/A | 1 2 3 4 | Please enter an array | Please enter an array | Pass |
| 3 | Empty Array B | 1 2 3 4 | N/a | Please enter an array | Please enter an array | Pass |
| 4 | Letter in Array A | 1 2 3 a | 1 2 3 4 5 | Please enter numbers only | Please enter numbers only | Pass |
| 5 | Letter in Array B | 1 2 3 4 5 | 1 2 3 a | Please enter numbers only | Please enter numbers only | Pass |
| 6 | Array A unsorted | 1 3 2 4 | 1 2 3 4 5 | Please enter sorted array | Please enter sorted array | Pass |
| 7 | Array B unsorted | 1 2 3 4 5 | 1 3 2 4 | Please enter sorted array | Please enter sorted array | Pass |
| 8 | Negative Inputs | -5 -3 4 5 | -3 -2 1 2 | [-5, -3, -3, -2, 1, 2, 5] | [-5, -3, -3, -2, 1, 2, 5] | Pass |
| 9 | Negative Inputs Out of Order | -1 -2 -3 -4 | -1 -2 -3 -4 | Please enter sorted array | Please enter sorted array | Pass |

**(Part III: 1) Data Flow Diagram**

**(PART III: 2-3) Independent Paths**

(1-2-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete simple path

(1-2-3-4-5-6-14-17-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Path

(1-2-3-4-5-6-7-8-15-17-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Path

(1-2-3-4-5-6-7-8-9-10-11-12-16-17-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Path

(1-2-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-29-32-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Simple Path

(1-2-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-30-32-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Path

(1-2-3-4-5-6-7-8-9-10-11-12-13-18-19-20-21-22-23-30-32-18-19-20-21-22-23-24-25-26-27-31-32-18-19-20-21-22-23-24-25-26-27-28-33-34-35-36-37) Complete Path

**(Part III: 4) Definition, C-use, P-use**

|  |  |  |
| --- | --- | --- |
| **Definition** | **C-use** | **P-use** |
| CheckA(42)  CheckB(43) |  |  |
|  |  | checkA(49) |
| userInput(50) |  |  |
| arrayA(51) | userInput(51) |  |
| checkA(52) | arrayA(52) |  |
|  |  | userInput(10) |
| num(33) | userInput(33) | num(33) |
|  |  | userInput(34)  num(34) |
| userInput(35) |  | num(35) |
|  |  | checkB(54) |
| userInput(55) |  |  |
| arrayB(56) | userInput(56) |  |
| checkB(57) | arrayB(57) |  |
|  |  | userInput(10) |
| num(33) | userInput(33) | num(33) |
|  |  | userInput(34)  num(34) |
| userInput(35) |  | num(35) |
| sortedArray(57) | arrayA(59)  arrayB(59) |  |
|  | sortedArray(60) |  |

Note: numbers here reference the program from Part I

**(Part III: 5) Def-Use Associations**

|  |  |  |
| --- | --- | --- |
| Variable Name | Defined at | Used at |
| checkA | 2 | (3, t), (3,f), 6 |
| checkB | 2 | (18, t), (18, f), 21 |
| userInput | 4 | 5, 19, 20 |
| arrayA | 5 | 6, (6, t), (6, f), 7, (8, t), (8, f), 9, (12, t), (12, f), 34, 35, 36, 37 |
| arrayB | 20 | 21, (21, t), (21, f), 22, (23, t), (23, f), 24, (27, t), (27, f), 34, 35 |
| sortedArray | 34 | 34 |
| checkDig | 7 | (11, t), (11, f), 22, (26, t), (26, f) |
| num | 8 | (8, t), (8, f), 9, (23, t), (23, f), 24 |

Note: numbers here reference the numbered data flow diagram

alternative format to follow lecture example:

(checkA, 2, (3, t)), (checkA, 2, (3, f)), (checkA, 2, 6)

(checkB, 2, (18, t)), (checkB, 2, (18, f)), (checkB, 2, 21)

(userInput, 4, 5), (userInput, 4, 19), (userInput, 4, 20)

(arrayA, 5, 6), (arrayA, 5, (6, t)), (arrayA, 5, (6, f)), (arrayA, 5, 7), (arrayA, 5, (8, t)), (arrayA, 5, (8, f)), (arrayA, 5, 9), (arrayA, 5, (12, t)), (arrayA, 5, (12, f)), (arrayA, 5, 34), (arrayA, 5, 35), (arrayA, 5, 36), (arrayA, 5, 37)

(arrayB, 20, 21), (arrayB, 20, (21, t)), (arrayB, 20, (21, f)), (arrayB, 20, 22), (arrayB, 20, (23, t)), (arrayB, 20, (23, f)), (arrayB, 20, 24), (arrayB, 20, (27, t)), (arrayB, 20, (27, f)), (arrayB, 20, 34), (arrayB, 20, 35)

(sortedArray, 34, 34)

(checkDig, 7, (11, t)), (checkDig, 7, (11, f)), (checkDig, 7, 22), (checkDig, 7, (26, t)), (checkDig, 7, (26, f))

(num, 8, (8, t)), (num, 8, (8, f)), (num, 8, 9), (num, 8, (23, t)), (num, 8, (23, t)), (num, 8, 24)