

DEPARTMENT OF MASTER OF

COMPUTER APPLICATION

Mathematical Foundation for Computer Applications

Activity - 2

**Name: Varanasi Surya Teja Application no.: JUPG22MCA12186 Semester: I Semester**

**Branch: MCA-AI & ML**

**Submitted to: Aravind Nagaraju**

**Professor**

**School of CS & IT**

**Question:**

**Write a Program Given a positive integer n, determine the number of equivalence relations on a set with n elements.**

**Explanation:**

**The program is about to get the equivalence relations by giving the conditions of reflexive, symmetric, transitive. By giving the user inputs the value of elements and enter the values. Later on it will generate the order pairs regarding to the given elements.**

**Code:**

**print("\n### Program to find Equivalence relation for a set ###\n")**

**l = []**

**#Get user inputs**

**num = int(input("Enter the number of elements: "))**

**for e in range(num):**

**l.append(int(input("Enter the element: ")))**

**#Enter your ordered pair into the orderedPair list and comment the ordered pair generator.**

**orderedPair = []**

**#### ORDERED PAIR GENERATOR ####**

**for m in range(len(l)):**

**for n in range(len(l)):**

**pair = (l[m], l[n])**

**orderedPair.append(pair)**

**####**

**# Print ordered pair**

**print("Generated Ordered pair: ",orderedPair)**

**def CheckReflexive(): # Check if relation is reflexive**

**for pair in orderedPair:**

**if (pair[0] == pair[1]):**

**print ("IS reflexive: ✓", pair, (pair[0], pair[1]))**

**break**

**else:**

**print("IS reflexive: ✕")**

**break**

**def CheckSymmetric(): # Check if relation is symmetric**

**for pair in orderedPair:**

**temp = (pair[1], pair[0])**

**if temp in orderedPair:**

**print ("IS symmetric: ✓", pair, (pair[1], pair[0]))**

**break**

**else:**

**print("IS symmetric: ✕")**

**break**

**def CheckTransitive(): # Check if relation is transitive**

**found = False**

**for pair in orderedPair:**

**for y in orderedPair:**

**if pair[1] == y[0] and not found:**

**temp = (pair[0], y[1])**

**if temp in orderedPair:**

**print("IS transitive: ✓", pair, y, temp)**

**found = True**

**break**

**else:**

**break**

**if orderedPair.index(pair) == len(orderedPair)-1 and not found:**

**print("IS transitive: ✕")**

**def TestReflexive(): # Remove all reflexive pairs**

**for p in orderedPair:**

**if p[0] == p[1]:**

**orderedPair.remove(p)**

**def TestSymmetric(): # Remove all symmetric pairs**

**TestReflexive()**

**for pair in orderedPair:**

**temp = (pair[1], pair[0])**

**if temp in orderedPair:**

**orderedPair.remove(pair)**

**def TestTransitive(): # Remove all transitive pairs**

**TestReflexive()**

**for pair in orderedPair:**

**for y in orderedPair:**

**if pair[1] == y[0]:**

**temp = (pair[0], y[1])**

**if temp in orderedPair:**

**orderedPair.remove(temp)**

**#TestReflexive() # uncomment line for testing**

**CheckReflexive()**

**#TestSymmetric() # uncomment line for testing**

**CheckSymmetric()**

**#TestTransitive() # uncomment line for testing**

**CheckTransitive()**

**print("Result ordered pair: ", orderedPair)**

**Output:**

**Enter the number of elements: 3**

**Enter the element: 1**

**Enter the element: 2**

**Enter the element: 3**

**Generated Ordered pair: [(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]**

**IS reflexive: ✓ (1, 1) (1, 1)**

**IS symmetric: ✓ (1, 1) (1, 1)**

**IS transitive: ✓ (1, 1) (1, 1) (1, 1)**

**Result ordered pair: [(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]**