Problem-1 : Let A be the set $\{1,2,3,4\}$. Write a program to find the ordered pairs in the relation $R_1 = \{(a,b) \mid a \text{ divides } b\}$ $R_2 = \{a,b \mid a \leq b\}$

Ans :

Theory: Understanding relations R_1 and R_2, on set $A = \{1,2,3,4\}$

In this theory, we will explore the two relations involves ordered pairs (a,b) based on certain conditions. Let's examine each relation seperately:

1. Relation R_1 : Here $R_1 = \{a,b \mid a \text{ divides } b\}$ relation R_1 involves ordered pairs (a,b) where 'a' divides 'b'. For the set $A = \{1,2,3,4\}$, the relation R_1 can be defined as follows.

$\therefore R_1 = \{(1,1), (1,2), (1,3), (1,4), (2,2), (2,4), (3,3) (4,4)\}$

2. Relation R_2: Here $R_2 = \{(a,b) \mid a \leq b\}$ Relation R_2 involves ordered pairs (a,b) where 'a' is less than or equal to 'b'. For the set the relation R_2 is:

$R_2 = \{(1,1)(1,2)(1,3)(1,4)(2,2)(2,3)(2,4)(3,3)(3,4)(4,4)\}$

Now the python program is written below:

Code:

```
from itertools import product
# Import the 'product function from the 'itertools' module.
With open("D:/python/Lab-01/input.txt", "r",
  encoding="utf-8") as f:
# Open the file 'input.txt' located in the "D:python/Lab-01/input.txt" directory in read mode and the file contains a list of integers.
  S = list(map(int, f, readlines()))
# Each line is read and converted to an integer, creating a list names 'S'.
```

Problem-02 : Suppose that $A=\{1,2,3\}$ and $B=\{1,2\}$, Let R be the relation from A to B containing (a,b) if $a \in A$, $b \in B$ and $a>b$. Write a program to find the relation R and also represent this relation in matrix form.

Ans:

Theory: In this problem, we are give two sets, A and B and we are asked to find the relation R from set A to set B based on specific condition The relation R will contain pairs (a,b) where 'a' belongs to set A, b belongs to set B and 'a' is greater than 'b'. In other words, the relation R will consist of all elements from set A that are greater than elements from set B.

The relation R is defined as follows: $R=\{(a,b) \mid a \in A, b \in B, a>b\}$. By compairing each element from set A with set B to check if 'a' is greater than 'b' then we will form the pair (a,b) of a set.

After forming the relation we need to assign the value of a pair (a, b) as 1 as a index of a matrix. $M_{(2,1)}$ means the index on row 2nd column 1 will have a value of 1 and other index will remain 0.

Now The python program is written below:

Code:

```
import numpy as np
# Importing the numpy library with alias 'np'
With open("m:/Vscode(Python)/Lab(ICE-2106)/LAB-REPORT
    -02/ input.txt","r",encoding="utf-8") as g:
# This file open a file named "input.txt" located in
the "m:/Vscode(Pytho)/Lab(ICE-2106)/Lab-Report-02/
directory in read mode with utf-8 encoding.
list1=list(map(int,g.readlines()))
# Read all lines from the file, convert each line to an
integer and store them in list 1
with open("m:/Vscode(python)\Lab(ICE-2106)/LAB-REPORT-01
/input.txt,"r",encoding="utf-8") as g:
# This line opens the same file "input.txt" again in
read mode ("r") with UTF-8 encoding.
```

Problem No : 03

Suppose that relation R_1 and R_2 on set A are represented by the matrics

$$M_{R1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } M_{22} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Write a program to find the $M_{R1} \cup R_2$ and $M_{R_1} \oplus R_2$.

Theory : The program shown above defines two function 'matrix_union()' and 'matrix exclusive_or(), to compute the union and symmetric union (exclusive or) of two matrics respectively.

The matrix_union() : Function takes two matrics as input and returns a new matrics that represented the union of two input matrics. It uses list comperi-hension to perform element-wise logical OR operation between the element of the two input matrics.

The matrix_exclusive or () function also takes two matrices an input and return a new matrices that represent the symmetric difference of the two input matrices. It uses list comperihension and the X-OR operator (^) to perform element wise logical X-OR operation between the elements of the two input matrices.

After defining this functions, the program creats two matrics R_1 and R_2 which represent relations. These matrix are then passed to the matrix union() and matrix_exclusive or() functions to compute the union and symmetric difference $MR_1 \oplus R_2$.

- Here a python program is written below:

```
def matrix union (matrix 1, matrix 2);
union result = [matrix 1 [i] [j] or matrix 2 [i] [j]
for j in range (len (matrix 1 [0]))]
for i in range (len ( matrix 1))]
return union_ result
```

Problem-04: Write a program to find shortest path by warshall's algorithm.

Ans:

Theory: The program shown above applies the Floyd-Warshall algorithm to find the shortest paths in a graph. The graph is represented by an adjacency matrix. The function "floyd-warshall()" takes two algorith: The number of vertics in the graph and the adjacency matrix representing the graph. Within this function, the Floyd warshall algorithm is implemented using nested for loop.

After applying the floyd warshall algorithm, the program prints the resulting adjacency matrix, showing the distance between each pair of vertics. The left row represent the origin vertics and the top most column represents the destination vertex. The values in the matrix represent the shortest distance between the corresponding vertex.

In the main function the number of vertices is set to 4. The adjacency matrix is defined based the given graph.

Finally the main function calls the Floyd warshall() matrix as the number of vertices and the adjacency matrix as arguments to find the shortest path in the graph.

Here a python program is written below:

code:

```
INF = 1000000000
# This line sets the value of "INF" to 1000000000, which is used to represent an infinite distance between vertics in the adjacency matrix.
def floyd_warshall (Vertex, adjacency_matrix);
# This line defines a function called "floyd_warshall" that takes two inputs "vertex" and "adjecaney-matrix".
for k in range (0, vertex);
# This line starts a loop that interates over the range of the vertex and use k as an intermediate vertex to find shortest path
```

Problem No:-05: Write a program for the solution of graph coloring by Welch-Powells.

Ans:

Theory: The graph coloring problem is classic optimization problem in computer science and graph theory. The objective is to assign colors to the vertices of a graph such that no two adjacent vectors share the same color. The Welch-Powell algorithm is a heurisite approch to solving this problem effectively. This lab report present implementation of Welch-Powell's algorithm to solve the graph coloring problem.

The Welch-Powell algorithm works as following:

1. start with an uncolored graph.
2. Iterate through the vertics and color each vertics with the lowest- possible color that is not used by its neighbors.
3. Repeat untill all vertics are colored.

Experiment Name: Find the root of the equation $x^2 - 4x - 10 = 0$ using bisection method.

1. Decide initial values for x_1 and x_2 and stopping criterion, E.
2. Compute $f_1 = f(x_1)$ and $f(x_2) = f_2$.
3. If $f_1 \times f_2 > 0$, x_1 and x_2 donot bracket any root and go to step 7.
4. Compute $x_0 = (x_1 + x_2)/2$ and compute $f_0 = f(x_0)$
5. If $f_1 \times f_0 < 0$ then

   Set $x_2 = x_0$

   else,

   Set $x_1 = x_0$

   Set $f_1 = f_0$
6. If absolute value of $(x_2 - x_1)/x_2$ is less than error E, then root $= (x_1 + x_2)/2$

   write the value of root

   go to step 7

   else go to step 4.
7. Stop.

the equation,

$$x^2 - 4x - 10 = 0$$

the maximum absolute of the solution:

$$x_{max} = \sqrt{\left(\frac{-4}{1}\right)^2 - 2\left(\frac{-10}{1}\right)}$$

$$= 6$$

the roots in the interval $(-6, 6)$. The table below gives the values of $f(x)$ between -6 and 6.

x	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
f(x)	50	35	22	11	2	-4	-10	-13	-14	-13	-10	-5	2

Let take $x_1 = -2$ and $x_2 = -1$

Then, $x_0 = \frac{-2-1}{2} = -1.5$

$f(-2) = 2$ and $f(-1.5) = -1.75$

Since $f(-2)\, f(-1.5) < 0$, the root must be in the interval $(-2, -1.5)$. The, $x_1 = -2, x_2 = 1.5$ and $x_0 = -1.625$

$f(-1.625) = -0.859$

Now, the root lies in the interval $(-1.75, -1.625)$ $x_0 = -1.6875$

$f(-1.6875) = 0.40$

Next, $x_0 = -\frac{-1.75 + 1.6875}{2} = -1.72$

$f(-1.72) = 0.1616$

Next, $x_0 = -\frac{1.75 + 1.75}{2} = -1.735$

$f(-1.735) = -0.05$

$x_0 = -1.7425$

$f(-1.7425) = 0.0063$

The root lies between -1.735 and 1.7425.

Approximate root is -1.7416.

Problem 02: To find the root of the $f(x) = x^2 - x - 2 = 0$ using the false position method. $[1 < x < 3]$

Algorithm:

Let,

$$x_0 = x_1 - f(x_1) \times \frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

If $f(x_0) \times f(x_1) < 0$

Set $x_2 = x_0$

Otherwise,

Set $x_1 = x_0$

Example:

Given that,

the equation, $f(x) = x^2 - x - 2 = 0$

$x = 1$ and $x_2 = 3$

$f(x_1) = f(1) = -2$

$f(x_2) = f(3) = 4$

Iteration 1:

$$x_0 = x_1 - f(x_1) \times \frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

$$= 1 + 2 \times \frac{3-1}{4+2}$$

$$= 1.6667$$

Iteration 2:

$$f(x_0) f(x_1) = f(1.6667) f(1)$$

$$= 1.7778$$

Therefore, the root lies in the interval between x_0 and x_2. Then, $x_1 = x_0 = 1.6667$

$f(x_1) = f(1.6667) = -0.8889$

$f(x_2) = f(3) = 4$

$$x_0 = 1.6667 + 0.8889 \times \frac{3 - 1.6667}{4 + 0.8889}$$

$$= 1.909$$

Iteration 03:

$f(1.909)\, f(1.6667) = +0.2345$

Root lies between $x_0 (= 1.909)$ and $x_2 (= 3)$

Therefore,

$$x_1 = x_0 = 1.909$$

$$x_3 = 3$$

$$x_0 = 1.909 + 0.2647 \times \frac{3 - 1.909}{4 - 0.2647}$$

$$= 1.909 + 0.2647 \times \frac{1.091}{3.7353}$$

$$= 1.986$$

The third root iteration is 1.986; the contain root interval $x = 2$. We can perform additional iteration to refine the estimate further.

Problem 03: To find the root of the equation $f(x) = x^2 - 3x + 2$ in the vicinity of $x = 0$ using Newton-Raphson method.

Algorithm:

1. Assign an initial value to x, say x_0.
2. Evaluate $f(x_0)$ and $f'(x_0)$
3. Find the improved estimate of x_0

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

4. Check for accuracy of the latest estimate.

Compare relative error to a predefined value E. If

$$\left| \frac{x_1 - x_0}{x_1} \right| \leq E \text{ Stop; otherwise continue.}$$

5. Replace x_0 by x_1 and repeat Steps 3 and 4.

Example:

the equation is, $f(x) = x^2 - 3x + 2$

$$f'(x) = 2x - 3$$

Let, $x_1 = 0$ (first approximation)

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$= 0 - \frac{2}{-3} = \frac{2}{3} = 0.6667$$

Similarly,

$$x_3 = 0.6667 - \frac{0.4444}{-1.6667} = 0.9333$$

$$x_4 = 0.9333 - \frac{0.071}{-1.334} = 0.9959$$

$$x_5 = 0.9959 - \frac{0.0041}{-1.0082} = 0.9999$$

$$x_6 = 0.9999 - \frac{0.0001}{-1.0002} = 1.0000$$

Since $f(1.0) = 0$; the root closest to the point $x = 0$ is 1.000.