2020

Assignment 3

DISCRETE STRUCTURES

HASSAM UD DIN | P190029 | BCS – 3A

File - 01:

userInput.java:

```
package Assignment3;

public class userInput {
    private int x;
    private int y;

    userInput(int x,int y)
    {
        this.x=x;
        this.y=y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }
    @Override
```

```
    public String toString()
    {
        String str= "(" + this.x + "," + this.y + ")";
        return str;
    }
}
```

File – 02:

Assignment3.java:

```
package Assignment3;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;

public class Assignment3 {

    public static boolean checkValid(ArrayList<Integer> A, ArrayList<userInput> R)
    {
        if (A.isEmpty())
        {
            if (R.isEmpty())
            {
                return true;
            }
            else
            {
                System.out.println("The input relation R is not valid for the set A");
                return false;
```

```
            }
        }
        for (int i=0;i<R.size();i++)
        {
            if (!A.contains(R.get(i).getX()) || !A.contains(R.get(i).getY()))
            {
                System.out.println("The input relation R is not valid for the set A");
                return false;
            }
        }
        return true;
    }

    public static boolean checkReflexive(ArrayList<Integer> A, ArrayList<userInput> R,boolean
printVar)
    {
        if (R.isEmpty() )
        {
            if (A.isEmpty())
            {
                return true;
            }
            return false;
        }
        int reqCount=A.size();
        int currCount=0;
        boolean foundFlag=false;
        for (int j=0;j<A.size();j++)
        {
            for (int i=0;i<R.size();i++)
```

```
            {
                if (A.get(j)==R.get(i).getX() && A.get(j)==R.get(i).getY())
                {
                    currCount++;
                    foundFlag=true;
                    break;
                }
            }
            if (foundFlag==false)
            {
                if (printVar)
                {
                    System.out.print("The following tuple was not found: " + "(" + A.get(j) + "," + A.get(j)
+ ")" );
                    return false;
                }
            }
            else
            {
                foundFlag=false;
            }
        }
        if (currCount==reqCount)
        {
            return true;
        }
        else
        {
            return false;
        }
```

```
    }


    public static boolean checkSymmetric(ArrayList<Integer> A,ArrayList<userInput> R,boolean
printVar)
    {
        if (R.isEmpty() )
        {
            return true;
        }
        int[] flags= new int[R.size()];
        for (int i=0;i<R.size();i++)
        {
            flags[i]=0;
        }
        for (int i=0;i<R.size();i++)
        {
            for (int j=0;j<R.size();j++)
            {
                if (R.get(i).getX()==R.get(j).getY() && R.get(i).getY()==R.get(j).getX())
                {
                    flags[i]=1;
                    break;
                }
            }
        }
        for (int i=0;i<flags.length;i++)
        {
            if (flags[i]==0)
            {
                if (printVar)
```

```
            {
                System.out.print("Following tuple was not found: " + "(" +
R.get(i).getY()+","+R.get(i).getX() + ")");
                return false;
            }


        }
    }
    return true;
}


public static boolean checkAntiSymmetric(ArrayList<Integer> A, ArrayList<userInput> R)
{
    boolean ret=false;
    if (R.size()>=A.size())
    {
      if (checkReflexive(A,R,false))
      {
          if (R.size()==A.size())
          {
              ret=true;
          }
          else if (checkSymmetric(A,R,false))
          {
              Random rand= new Random();
              int x= rand.nextInt(R.size());
              while (R.get(x).getX()==R.get(x).getY())
              {
                  x= rand.nextInt(R.size());
              }
```

```
                System.out.print( "(" + R.get(x).getX() + "," + R.get(x).getY() + ") " + "and " + "(" +
R.get(x).getY() + "," + R.get(x).getX() + ")" + " exist but " + R.get(x).getX() + "!=" +
R.get(x).getY() + "\t");
                ret=false;
            }
            else
            {
                ret= true;
            }
        }
        else
        {
            if (!checkSymmetric(A,R,false))
            {
                ret= true;
            }
            else
            {
                Random rand= new Random();
                int x= rand.nextInt(R.size());
                while (R.get(x).getX()!=R.get(x).getY())
                {
                    x= rand.nextInt(R.size());
                }
                System.out.print( "(" + R.get(x).getX() + "," + R.get(x).getY() + ")" + "and" + "(" +
R.get(x).getY() + "," + R.get(x).getX() + ")" + "exist but" + R.get(x).getX() + "!=" +
R.get(x).getY());
                ret=false;
            }
        }
    }
```

```
    else
    {
        if (!checkReflexive(A,R,false))
        {
            if (!checkSymmetric(A,R,false))
            {
                ret= true;
            }
            else
            {
                ret=true;
                for (int i=0;i<R.size();i++)
                {
                    if (R.get(i).getX()!=R.get(i).getY())
                    {
                        ret=false;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}

public static void main(String[] args) {
    ArrayList<Integer> A= new ArrayList<Integer>();
    ArrayList<userInput> R= new ArrayList<userInput>();
    A.add(1);
    A.add(2);
```

```
        A.add(3);
        A.add(4);
        A.add(5);
        A.add(6);
        System.out.println("A= "+A);
        R.add(new userInput(1,1));
        R.add(new userInput(2,2));
        //R.add(new userInput(2,3));
        //R.add(new userInput(3,2));
       // R.add(new userInput(3,3));
      //  R.add(new userInput(4,4));
      //  R.add(new userInput(5,5));
      //  R.add(new userInput(6,6));
        System.out.println("R= "+R);
        boolean valid= checkValid(A,R);
        if (valid)
        {
            System.out.println("Reflexive: " + checkReflexive(A,R,true));
            System.out.println("Symmetric: " + checkSymmetric(A,R,true));
            System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
            System.out.println();
        }
    }
}
```

OUTPUTS:

Test case 1:

Start Page | userInput.java | Assignment3.java

Source | History

```
                }
            }
        }
        return ret;
    }

    public static void main(String[] args) {
        ArrayList<Integer> A= new ArrayList<Integer>();
        ArrayList<userInput> R= new ArrayList<userInput>();
      //  A.add(1);
      //  A.add(2);
     //   A.add(3);
     ///    A.add(4);
    //     A.add(5);
    //     A.add(6);
        System.out.println("A= "+A);
        R.add(new userInput('a','a'));
        //R.add(new userInput(2,2));
        //R.add(new userInput(2,3));
       // R.add(new userInput(3,2));
       // R.add(new userInput(3,3));
      //  R.add(new userInput(4,4));
     //   R.add(new userInput(5,5));
```

Output - ChishtiChinoy (run)

```
run:
A= []
R= [(a,a)]
The input relation R is not valid for the set A
BUILD SUCCESSFUL (total time: 0 seconds)
```

Test case 2:

Start Page | userInput.java | Assignment3.java

Source | History

```
//   A.add(1);
//   A.add(2);
//   A.add(3);
///   A.add(4);
//    A.add(5);
//    A.add(6);
    System.out.println("A= "+A);
    //R.add(new userInput('a','a'));
    //R.add(new userInput(2,2));
    //R.add(new userInput(2,3));
   // R.add(new userInput(3,2));
   // R.add(new userInput(3,3));
  //  R.add(new userInput(4,4));
 //   R.add(new userInput(5,5));
//    R.add(new userInput(6,6));
    System.out.println("R= "+R);
    boolean valid= checkValid(A,R);
    if (valid)
    {
        System.out.println("Reflexive: " + checkReflexive(A,R));
        System.out.println("Symmetric: " + checkSymmetric(A,R));
        System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
        System.out.println();
```

Output - ChishtiChinoy (run)

```
run:
A= []
R= []
Reflexive: true
Symmetric: true
AntiSymmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Test case 3:

Start Page | userInput.java | Assignment3.java

Source | History

```java
            A.add(2);
            A.add(3);
            A.add(4);
            A.add(5);
            A.add(6);
            System.out.println("A= "+A);
            R.add(new userInput(1,1));
            R.add(new userInput(2,2));
            //R.add(new userInput(2,3));
            R.add(new userInput(3,2));
            R.add(new userInput(3,3));
            R.add(new userInput(4,4));
            R.add(new userInput(5,5));
            R.add(new userInput(6,6));
            System.out.println("R= "+R);
            boolean valid= checkValid(A,R);
            if (valid)
            {
                System.out.println("Reflexive: " + checkReflexive(A,R));
                System.out.println("Symmetric: " + checkSymmetric(A,R,true));
                System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
                System.out.println();
            }
```

Output - ChishtiChinoy (run)

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2), (3,2), (3,3), (4,4), (5,5), (6,6)]
Reflexive: true
Following tuple was not found: (2,3)Symmetric: false
Anti-Symmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Test case 4:

Start Page | userInput.java | Assignment3.java

Source | History

```java
        A.add(1);
        A.add(2);
        A.add(3);
        A.add(4);
        A.add(5);
        A.add(6);
        System.out.println("A= "+A);
        R.add(new userInput(1,1));
        R.add(new userInput(2,2));
        R.add(new userInput(2,3));
        R.add(new userInput(3,2));
        R.add(new userInput(3,3));
        R.add(new userInput(4,4));
        R.add(new userInput(5,5));
        R.add(new userInput(6,6));
        System.out.println("R= "+R);
        boolean valid= checkValid(A,R);
        if (valid)
        {
            System.out.println("Reflexive: " + checkReflexive(A,R));
            System.out.println("Symmetric: " + checkSymmetric(A,R,true));
            System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
            System.out.println();
        }
```

Output - ChishtiChinoy (run)

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2), (2,3), (3,2), (3,3), (4,4), (5,5), (6,6)]
Reflexive: true
Symmetric: true
(2,3) and (3,2) exist but 2!=3  Anti-Symmetric: false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Test case 5:

Start Page | userInput.java | Assignment3.java

Source | History

```java
A.add(1);
A.add(2);
A.add(3);
A.add(4);
A.add(5);
A.add(6);
System.out.println("A= "+A);
R.add(new userInput(1,1));
R.add(new userInput(2,2));
//R.add(new userInput(2,3));
//R.add(new userInput(3,2));
// R.add(new userInput(3,3));
//  R.add(new userInput(4,4));
//  R.add(new userInput(5,5));
//  R.add(new userInput(6,6));
System.out.println("R= "+R);
boolean valid= checkValid(A,R);
if (valid)
{
    System.out.println("Reflexive: " + checkReflexive(A,R,true));
    System.out.println("Symmetric: " + checkSymmetric(A,R,true));
    System.out.print("Anti-Symmetric: " + checkAntiSymmetric(A,R));
    System.out.println();
}
```

Output - ChishtiChinoy (run)

```
run:
A= [1, 2, 3, 4, 5, 6]
R= [(1,1), (2,2)]
The following tuple was not found: (3,3)Reflexive: false
Symmetric: true
Anti-Symmetric: true
BUILD SUCCESSFUL (total time: 0 seconds)
```