ASSIGNMENT-1

Q1)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix        iiitd-sweb: <http://www.iiitd.ac.in/course/sweb/>.
@prefix        assign: <http://www.iiitd.ac.in/course/sweb/assign/>.
@prefix        assignO: <http://www.iiitd.ac.in/course/sweb/assignO/>.
@prefix        eg: <http://www.iiitd.ac.in/example/>.
@prefix schemaorg: <http://schema.org/>.
```

(a)

```
assign:juiceMadeOfFruit rdf:type rdf:Property .
assign:juiceMadeOfFruit       rdfs:domain assign:FruitJuice .
assign:juiceMadeOfFruit       rdfs:range schemaorg:Fruit .
```

(b)

```
assign:Fruit rdf:type rdfs:Class .
assign:apple rdf:type assign:Fruit .
```

(c)

```
assign:juiceMadeOf rdf:type rdf:Property .
assign:juiceMadeOfFruit rdf:type rdf:Property .
assign:juiceMadeOfFruit rdfs:subPropertyOf assign:juiceMadeOf .
```

(d)

```
//Container of type "bag" is used because the order of elements
//do not matter for making the MixedFruitJuice(as per context given).

assign:mixedFruitJuice assign:isMadeOf [
                                    rdf:type rdf:Bag ;
                                    rdf:_1 <Banana> ;
                                    rdf:_2 <Orange> ;
                                    rdf:_3 <Pineapple> ;
                                    rdf:_4 <Watermelon> ;
                                ].
```

(e) **//3 blanknodes are used to represent information about**
//the given 3 ingredients.

```
assignO:mixedFruitJuice eg:isMadeOf _:item1.
assignO:mixedFruitJuice eg:isMadeOf _:item2.
assignO:mixedFruitJuice eg:isMadeOf _:item3.

_:item1
        eg:ingredient eg:Orange ;
        eg:amount "2"^^xsd:Integer .

_:item2
        eg:ingredient eg:Pomegranate ;
        eg:quantity "1"^^xsd:Integer .

_:item3
        eg:ingredient eg:Pineapple ;
        eg:quantity "1"^^xsd:Integer .
```

(f) **//2 blank nodes are used to represent information about**
//the given 2 ingredients.

```
assign:orangeJuice eg:isMadeOf
                                [ eg:ingredient eg:Orange ;
                                  eg:quantity "3"^^xsd:Integer ].
assign:orangeJuice eg:isMadeOf
                                [ eg:ingredient eg:salt ;
                                  eg:quantity "1 tablespoon"^^xsd:String ].
```

(g)

```
eg:FruitJuice rdf:type rdfs:Class .
eg:MixedFruitJuice rdf:type rdfs:Class .
eg:MixedFruitJuice rdfs:subClassOf eg:FruitJuice .
```

(h)

```
assignO:Fruit rdf:type rdfs:Class .
assignO:FruitJuice rdf:type rdfs:Class .
assignO:MixedFruitJuice rdf:type rdfs:Class .
```

(i)

```
eg:juiceMadeOf rdf:type rdf:Property .
eg:juiceMadeOfFruit rdf:type rdf:Property .
```

(j)

```
eg:glassOfJuice eg:costs "INR 25"^^xsd:string.
```

****

Q2)

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix        person: <http://www.iiitd.ac.in/person/>.
@prefix        prop: <http://www.iiitd.ac.in/property/>.
@prefix        eg: <http://www.iiitd.ac.in/example/>.

//Container of type "Seq" is used because Mary likes the
//items in a particular order.
person:Mary prop:likes [
                rdf:type rdf:Seq ;
                rdf:_1 <MixedFruitJuice> ;
                rdf:_2 <OrangeJuice> ;
                rdf:_3 <AppleJuice> ;
        ] .

//Container of type "Alt" is used when we have to select only
//one among the given entities.(or things).
//As Mary prefers only one among these items, we have used
//"Alt" here.
eg:duringFruitMeal prop:prefers [
                rdf:type rdf:Alt ;
                rdf:_1 <Orange> ;
                rdf:_2 <Apple> ;
                rdf:_3 <Pineapple> ;
        ] .
```

```
//Container of type "Bag" is used when the order of the items
//is not important.
//As the MixedFruitJuice can be prepared by adding the mentioned
//fruits in any order, we can use "Bag" container here.
eg:mixedFruitJuice prop:isMadeOf [
                rdf:type rdf:Bag ;
                rdf:_1 <Oranges> ;
                rdf:_2 <Apple> ;
                rdf:_3 <Papaya> ;
                rdf:_4 <Banana> ;
        ].
```

****

4)

```
import java.util.*;



import org.eclipse.rdf4j.model.Model;
import org.eclipse.rdf4j.model.Statement;
import org.eclipse.rdf4j.model.util.ModelBuilder;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintStream;
public class Final4 {

        public static void main(String[] args) throws FileNotFoundException {
                //setting the path of the file "Netflix.csv"
                String fileN = "C:\\CodeRep\\Sample1\\NetflixList.csv";
                //Storing properties in a constant String array
                String[] prop = new String[] {"show_id","type","title",
                                                        "Director","cast","country",

                                                "Date_added","release_year","rating",

                                                "duration","listed_in","description"
                                                };
```

```
File f = new File(fileN);
ModelBuilder builder = new ModelBuilder();
builder.setNamespace("sub", "http://a1Subject.org/");
builder.setNamespace("pro", "http://a1Property.org/");
builder.setNamespace("obj", "http://a1Object.org/");
//Reference for writing output to a file

//https://www.geeksforgeeks.org/redirecting-system-out-println-output-to-a-file-in-java/
PrintStream o = new PrintStream(new
File("C:\\Users\\hanum\\Desktop\\Q4_t2.ttl"));
System.setOut(o);
String data;
int i=0;
try {
        String cols[];
        Scanner sc = new Scanner(f);
        //Splitting into rows using "\n" as a delimitter
        sc.useDelimiter("\n");
        while(sc.hasNext()) {
                data = sc.next();
                //Splitting columns into seperate values
                //Reference for the regex

                //https://stackoverflow.com/questions/18893390/splitting-on-comma-outside-quotes
                cols=data.split(",(?=([^\"]*\"[^\"]*\")*[^\"]*$)");
                if(i>0)
                {
                        builder.subject("sub:"+cols[0]);
                        for(int j=1;j<cols.length;++j)
                        {

                                if((cols[j].compareTo("")!=0)&&(j!=3 &&
                                j!=4 && j!=5 && j!=10))
                        {
                                builder.add("pro:"+prop[j],"obj:"+cols[j]);
                        }
```

```
        //handling column values containing comma
        within the value

        else if((cols[j].compareTo("")!=0) &&
        (cols[j].contains(",")))
        {
                String[] mult_cols = cols[j].split(",");

                //Each such comma separated value
                becomes a new value
                for(int k=0;k<mult_cols.length;++k)
                {


                        builder.add("pro:"+prop[j],
                        "obj:"+mult_cols[k]);
                }
        }
        //handling values for which are empty
        else if(cols[j].compareTo("")!=0)
        {
                builder.add("pro:"+prop[j], "obj:"+cols[j]);
        }
}
builder.subject("sub:"+cols[2]);
//Creating triples with Subject as "title"
for(int j=3;j<cols.length;++j)
{

                        if((cols[j]!=null)&&(j!=3 && j!=4
                && j!=5 && j!=10))
        {
                builder.add("pro:"+prop[j],"obj:"+cols[j]);
        }
        else if((cols[j]!=null) && (cols[j].contains(",")))
        {
                String[] mult_cols = cols[j].split(",");
                for(int k=0;k<mult_cols.length;++k)
```

```
                                        {


                                                builder.add("pro:"+prop[j],
                                                "obj:"+mult_cols[k]);
                                        }
                                }
                                else if(cols[j].compareTo("")!=0)
                                {
                                        builder.add("pro:"+prop[j], "obj:"+cols[j]);
                                }
                        }
                }
                i++;

        }
        Model model = builder.build();
        for(Statement st: model) {
                System.out.println(st);
         }
        sc.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }

}
}
```