CSE632 Semantic Web

Assignment - 1

Name- **Kaustav Vats**
Roll No- **2016048**

Question 1

```
@prefix : <http://www.iiitd.ac.in/sweb/a1/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#> .
```

A.
```
:juiceMadeOfFruit
        rdf:type        rdf:Property ;
        rdfs:domain     :FruitJuice ;
        rdfs:range      :Fruit .
```
B.
```
:Apple          rdf:type        :Fruit.
```
C.
```
:juiceMadeOfFruit   rdf:subPropertyOf   :juiceMadeOf
```
D.
```
:MixedFruitJuice    :hasIngredients     :Banana, :Orange, :Pineapple, :Watermelon .
```
E.
```
:MixedFruitJuice    :hasIngredients     [
                :Fruit          :Orange ;
                :Quantity       "2"^^xsd:integer
        ],      [
                :Fruit          :Pomegranate ;
                :Quantity       "1"^^xsd:integer
        ],      [
                :Fruit          :Pineapple ;
                :Quantity       "1"^^xsd:integer
        ] .
```
F.
```
:OrangeJuice        :hasIngredients     [
                :Fruit          :Orange ;
                :Quantity       "3"^^xsd:integer
        ],      [
                :Other          :TablespoonOfSalt ;
                :Quantity        "1"^^xsd:integer
        ] .
```
G.
```
:MixedFruitJuice    rdfs:subClassOf     :FruitJuice.
```
H.
```
:Fruit              rdf:type        rdfs:Class.
```

:FruitJuice rdf:type rdfs:Class.
:MixedFruitJuice rdf:type rdfs:Class.

I. :juiceMadeOfFruit rdf:type rdf:Property.
:juiceMadeOf rdf:type rdf:Property.

J. :juice :cost [
:glassCount "1"^^xsd:integer ;
rdf:value "25"^^xsd:integer ;
:currency "INR"^^xsd:string
] .

Question - 2

```
@prefix : <http://www.iiitd.ac.in/sweb/a1/q2/> .
@prefix property: <http://www.iiitd.ac.in/sweb/a1/q2/property#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

property:Likes a rdf:Property ;
    rdfs:domain  rdfs:Resource ;
    rdfs:range   rdfs:Resource .

property:madeOf a rdf:Property ;
    rdfs:domain  rdfs:Resource ;
    rdfs:range   rdfs:Resource .

property:Preference a rdf:Property ;
    rdfs:domain  rdfs:Resource ;
    rdfs:range   rdfs:Resource .

property:FruitMealPreference  rdf:subPropertyOf property:Preference ;
    rdfs:domain  rdfs:Resource ;
    rdfs:range   rdfs:Literal .

:FruitJuice     rdf:type    rdfs:Class .

:OrangeJuice    rdfs:subClassOf    :FruitJuice .
:AppleJuice     rdfs:subClassOf    :FruitJuice .
:MixedFruitJuice    rdfs:subClassOf    :FruitJuice .
```

```
:Mary a foaf:Person ;
    property:Likes [
        a rdf:Bag ;
        rdf:_1  :OrangeJuice ;
        rdf:_2  :AppleJuice ;
        rdf:_3  :MixedFruitJuice
    ] ;
    property:Preference [
        a rdf:Seq ;
        rdf:_1  :MixedFruitJuice ;
        rdf:_2  :OrangeJuice ;
        rdf:_3  :AppleJuice
    ] ;
    property:FruitMealPreference [
        a rdf:Alt ;
        rdf:_1  :Pineapple ;
        rdf:_2  :Orange ;
        rdf:_3  :Apple
    ] .

:MixedFruitJuice    property:madeOf     _:item1 .
_:item1
    rdf:first   :Orange ;
    rdf:rest    _:item2 .
_:item2
    rdf:first   :Apple ;
    rdf:rest    _:item3 .
_:item3
    rdf:first   :Papaya ;
    rdf:rest    _:item4 .
_:item4
    rdf:first   :Banana ;
    rdf:rest    rdf:nil .
```

1. I used Bag to list the likes of Juices. Bag container is used to describe a list of items that are intended to be unordered. Since Mary likes Orange Juice, Apple Juice and Mixed Fruit Juice, we can directly add all 3 to the same set of liked juices without any specific order.
2. For preference order of juices, I used Seq container. It is used to describe a list of items that are intended to be ordered.

3. For fruit meal preference, I used Alt container. It is used to describe a list of alternate values. Only one value can be selected from Alt container.
4. For mixed fruit juice that is made of **only** Orange, Apple, Papaya and Banana. I used RDF Collections because they are used to describe groups that contain only the specified items. In our case we were describing “which fruits are used to make mixed fruit juice”. Using RDF Collections i have listed only those fruits which are used to make mixed fruit juice.

Question 3

Tool used for visualization - RDF Data Visualization

Uses rdflib and Graphviz Library to create svg graphs.

Created a relation between start node and end node using RDF Triples.

Question 4

- Handled empty cell (by length etc.)
- Created custom uri, by removing special characters except space and replacing space with underscore.

Classes:

- Movie and TV Show are a Class
- Node containing whole data itself is a type of Movie or a TV Show.
- All Cast and Directors are type of Person class and also Person itself is a Class.

Properties:

- hasTitle
- countries
- hasRating
- listedIn
- releaseYear
- duration
- description
- dateAdded

I have also used some predefined properties like rdf: label from the Vocabulary. Domain and Range of all the properties are listed in TTL file.

All codes are available in Code folder.

All functions are self explanatory. For any doubt in any function, please call me for viva.

```
public class Q3 {
    public Model model;
    public String id;
    public String custom_uri;
```

```
    public String node_url, rdf;


    public Q3() {
        this.model = ModelFactory.createDefaultModel();
        this.id = null;
        this.custom_uri = "http://www.iiitd.ac.in/winter2020/sweb/a1/";
        this.node_url = "http://api.conceptnet.io";
        this.rdf = "http://www.w3.org/2000/01/rdf-schema#";
    }

    public String hit(String link) {
        try {
            URL url = new URL(link);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Accept", "application/ld+json");

            if (conn.getResponseCode() != 200) {
                throw new RuntimeException("Failed : HTTP error code : " +
conn.getResponseCode());
            }

            BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

            String result = "";
            String output;
            while ((output = br.readLine()) != null) {
                result += output;
            }
            conn.disconnect();
            return result;
        }
        catch (Exception e) {
            e.printStackTrace();
            return "";
        }
    }
```

```
    public String getNextUrl(String jData) {
        try {
            JSONParser parser = new JSONParser();
            JSONObject jObject = (JSONObject) parser.parse(jData);
            if (jObject.containsKey("view") ) {
                Map view = (Map) jObject.get("view");
                if (view.containsKey("nextPage")) {
                    return "http://api.conceptnet.io" + view.get("nextPage");
                }
            }
            return "";
        }
        catch (ParseException e) {
            e.printStackTrace();
            return "";
        }
    }

    public void parseJSONLD(String url) {
        try {
            String jData = hit(url);
            JSONParser parser = new JSONParser();
            JSONObject jObject = (JSONObject) parser.parse(jData);
            if (jObject.containsKey("error")) {
                Map error = (Map) jObject.get("error");
                if (error.containsKey("details")) {
                    System.out.println("Error: " + error.get("details"));
                    return;
                }
            }
            this.id = (String) jObject.get("@id");
            JSONArray jArray = (JSONArray) jObject.get("edges");
//            System.out.println(jArray);

            if (jArray.size() == 0) {
                return;
            }
            for (int i=0; i<jArray.size(); i++) {
                JSONObject tempObject = (JSONObject) jArray.get(i);
                Map rel = (Map) tempObject.get("rel");
```

```
                Property tempProperty = this.model.createProperty(custom_uri,
String.valueOf(rel.get("label")));
                tempProperty.addProperty(RDFS.domain, RDFS.Resource);
                tempProperty.addProperty(RDFS.range, RDFS.Resource);

                Map Node1 = (Map) tempObject.get("start");

                Property label = this.model.createProperty(custom_uri, "label");
                label.addProperty(RDFS.domain, RDFS.Resource);
                label.addProperty(RDFS.range, RDFS.Literal);
                Property language = this.model.createProperty(custom_uri,
"language");
                language.addProperty(RDFS.domain, RDFS.Resource);
                language.addProperty(RDFS.range, RDFS.Literal);

                Resource Start = this.model.createResource(node_url +
String.valueOf(Node1.get("@id")));
                Start.addProperty(label, String.valueOf(Node1.get("label")));
                Start.addProperty(language,
String.valueOf(Node1.get("language")));

                Map Node2 = (Map) tempObject.get("end");
                Resource End = this.model.createResource(node_url +
String.valueOf(Node2.get("@id")));
                End.addProperty(label, String.valueOf(Node2.get("label")));
                End.addProperty(language, String.valueOf(Node2.get("language")));

                Start.addProperty(tempProperty, End);
            }
            this.model.setNsPrefix("custom", this.custom_uri);
            this.model.setNsPrefix("rdf", this.rdf);
            Writer out = new OutputStreamWriter(new
FileOutputStream("output.ttl"), StandardCharsets.UTF_8);
            this.model.write(out, "TTL");
            out.close();
        }
        catch (ParseException | IOException e) {
            e.printStackTrace();
        }
    }
```

```
    public static void main(String[] args) throws IOException {
        System.out.print("Enter word: ");
        Q3 q3 = new Q3();
        BufferedReader Reader = new BufferedReader(new
InputStreamReader(System.in));
        String word = Reader.readLine();
        String link = "http://api.conceptnet.io/c/en/";
        q3.parseJSONLD(link + word);
        System.out.println("Output stored in output.ttl");
    }
}
```

Question 4 Code

```
public class NetflixObject {
    public String show_id;
    public String type;
    public String title;
    public List<String> directors;
    public List<String> cast;
    public List<String> countries;
    public String date_added;
    public String release_year;
    public String rating;
    public String duration;
    public List<String> listed_in;
    public String description;

    public void setShow_id(String show_id) {
        this.show_id = show_id;
    }

    public void setType(String type) {
        this.type = type;
    }

    public void setTitle(String title) {
        this.title = title;
    }
```

```java
public void setDirectors(String[] directors) {
    this.directors = Arrays.asList(directors);
}

public void setCast(String[] cast) {
    this.cast = Arrays.asList(cast);
}

public void setCountries(String[] countries) {
    this.countries = Arrays.asList(countries);
}

public void setDate_added(String date_added) {
    this.date_added = date_added;
}

public void setRelease_year(String release_year) {
    this.release_year = release_year;
}

public void setRating(String rating) {
    this.rating = rating;
}

public void setDuration(String duration) {
    this.duration = duration;
}

public void setListed_in(String[] listed_in) {
    this.listed_in = Arrays.asList(listed_in);
}

public void setDescription(String description) {
    this.description = description;
}

public String getShow_id() {
    return show_id;
}
```

```java
public String getType() {
    return type;
}

public String getTitle() {
    return title;
}

public List<String> getDirectors() {
    return directors;
}

public List<String> getCast() {
    return cast;
}

public List<String> getCountries() {
    return countries;
}

public String getDate_added() {
    return date_added;
}

public String getRelease_year() {
    return release_year;
}

public String getRating() {
    return rating;
}

public String getDuration() {
    return duration;
}

public List<String> getListed_in() {
    return listed_in;
}
```

```
    public String getDescription() {
        return description;
    }

    @Override
    public String toString() {
        return "NetflixObject{" +
                "show_id='" + show_id + '\'' +
                ", type='" + type + '\'' +
                ", title='" + title + '\'' +
                ", directors=" + directors +
                ", cast=" + cast +
                ", countries=" + countries +
                ", date_added='" + date_added + '\'' +
                ", release_year='" + release_year + '\'' +
                ", rating='" + rating + '\'' +
                ", duration='" + duration + '\'' +
                ", listed_in='" + listed_in + '\'' +
                ", description='" + description + '\'' +
                '}';
    }
}
```

```
public class csvNetflixParser {
    public ArrayList<NetflixObject> parse(String filepath) {
        try {
            FileReader filereader = new FileReader(filepath);
            CSVReader csvReader = new CSVReader(filereader);
            String[] nextRecord;
            ArrayList<NetflixObject> Arr = new ArrayList<>();
            while ((nextRecord = csvReader.readNext()) != null)
            {
                NetflixObject temp = new NetflixObject();
                temp.setShow_id(nextRecord[0]);
                temp.setType(nextRecord[1]);
                temp.setTitle(nextRecord[2]);
                temp.setDirectors(nextRecord[3].split(", "));
```

```
                temp.setCast(nextRecord[4].split(", "));
                temp.setCountries(nextRecord[5].split(", "));
                temp.setDate_added(nextRecord[6]);
                temp.setRelease_year(nextRecord[7]);
                temp.setRating(nextRecord[8]);
                temp.setDuration(nextRecord[9]);
                temp.setListed_in(nextRecord[10].split(", "));
                temp.setDescription(nextRecord[11]);
                Arr.add(temp);
            }
            Arr.remove(0);
            return Arr;
        }
        catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

```
public class csvToTriples {
    public Model model;
    public String properties_uri;
    public String node_url;
    public String rdf, rdfProperty;
    public Property hasTitle, hasDirector, hasCast, countries, hasRating,
listedIn, description, dateAdded, releaseYear, duration;
    public Resource person;

    public csvToTriples() {
        this.model = ModelFactory.createDefaultModel();
        this.properties_uri = "http://netflix.io/property/";
        this.node_url = "http://netflix.io/node/";
        this.rdf = "http://www.w3.org/2000/01/rdf-schema#";
        this.rdfProperty = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
        this.person = this.model.createResource(node_url + "Person");
        this.person.addProperty(RDF.type, RDFS.Class);

        this.hasTitle = this.model.createProperty(properties_uri, "hasTitle");
        this.hasTitle.addProperty(RDF.type, RDF.Property);
```

```
        this.hasTitle.addProperty(RDFS.domain, RDFS.Resource);
        this.hasTitle.addProperty(RDFS.range, RDFS.Literal);

        this.hasDirector = this.model.createProperty(properties_uri,
"hasDirector");
        this.hasDirector.addProperty(RDF.type, RDF.Property);
        this.hasDirector.addProperty(RDFS.domain, RDFS.Resource);
        this.hasDirector.addProperty(RDFS.range, this.person);

        this.hasCast = this.model.createProperty(properties_uri, "hasCast");
        this.hasCast.addProperty(RDF.type, RDF.Property);
        this.hasCast.addProperty(RDFS.domain, RDFS.Resource);
        this.hasCast.addProperty(RDFS.range, this.person);

        this.countries = this.model.createProperty(properties_uri, "countries");
        this.countries.addProperty(RDF.type, RDF.Property);
        this.countries.addProperty(RDFS.domain, RDFS.Resource);
        this.countries.addProperty(RDFS.range, RDFS.Literal);

        this.hasRating = this.model.createProperty(properties_uri, "hasRating");
        this.hasRating.addProperty(RDF.type, RDF.Property);
        this.hasRating.addProperty(RDFS.domain, RDFS.Resource);
        this.hasRating.addProperty(RDFS.range, RDFS.Literal);

        this.listedIn = this.model.createProperty(properties_uri, "listedIn");
        this.listedIn.addProperty(RDF.type, RDF.Property);
        this.listedIn.addProperty(RDFS.domain, RDFS.Resource);
        this.listedIn.addProperty(RDFS.range, RDFS.Literal);

        this.description = this.model.createProperty(properties_uri,
"description");
        this.description.addProperty(RDF.type, RDF.Property);
        this.description.addProperty(RDFS.domain, RDFS.Resource);
        this.description.addProperty(RDFS.range, RDFS.Literal);

        this.dateAdded = this.model.createProperty(properties_uri, "dateAdded");
        this.dateAdded.addProperty(RDF.type, RDF.Property);
        this.dateAdded.addProperty(RDFS.domain, RDFS.Resource);
        this.dateAdded.addProperty(RDFS.range, RDFS.Literal);
```

```
        this.releaseYear = this.model.createProperty(properties_uri,
"releaseYear");
        this.releaseYear.addProperty(RDF.type, RDF.Property);
        this.releaseYear.addProperty(RDFS.domain, RDFS.Resource);
        this.releaseYear.addProperty(RDFS.range, RDFS.Literal);

        this.duration = this.model.createProperty(properties_uri, "duration");
        this.duration.addProperty(RDF.type, RDF.Property);
        this.duration.addProperty(RDFS.domain, RDFS.Resource);
        this.duration.addProperty(RDFS.range, RDFS.Literal);
    }

    public void convertToTTL(String filepath) throws IOException {
        csvNetflixParser csvNetflixParser = new csvNetflixParser();
        ArrayList<NetflixObject> Arr = csvNetflixParser.parse(filepath);
        List<String> temp;


        for (NetflixObject e : Arr) {
            Resource movie = this.model.createResource(node_url + e.getShow_id());

            // Type
            Resource rType = this.model.createResource(node_url +
createURI(e.getType()));
            rType.addProperty(RDFS.label, e.getType());
            rType.addProperty(RDF.type, RDFS.Class);
            movie.addProperty(RDF.type, rType);

            movie.addProperty(this.hasTitle, e.getTitle());
            movie.addProperty(this.dateAdded, e.getDate_added());
            movie.addProperty(this.releaseYear, e.getRelease_year());
            movie.addProperty(this.duration, e.getDuration());
            movie.addProperty(this.description, e.getDescription());
            movie.addProperty(this.hasRating, e.getRating());

            temp = e.getListed_in();
            for (int i=0; i<temp.size(); i++) {
                movie.addProperty(this.listedIn, temp.get(i));
            }
```

```
            temp = e.getDirectors();
            for (int i=0; i<temp.size(); i++) {
                if (temp.get(i).length() == 0) {
                    continue;
                }
                Resource rUser = this.model.createResource(node_url +
createURI(temp.get(i)));
                rUser.addProperty(RDFS.label, temp.get(i));
                rUser.addProperty(RDF.type, RDFS.Resource);
                movie.addProperty(this.hasDirector, rUser);
            }

            temp = e.getCast();
            for (int i=0; i<temp.size(); i++) {
                if (temp.get(i).length() == 0) {
                    continue;
                }
                Resource rUser = this.model.createResource(node_url +
createURI(temp.get(i)));
                rUser.addProperty(RDFS.label, temp.get(i));
                rUser.addProperty(RDF.type, RDFS.Resource);
                movie.addProperty(this.hasCast, rUser);
            }

            temp = e.getCountries();
            for (int i=0; i<temp.size(); i++) {
                if (temp.get(i).length() == 0) {
                    continue;
                }
                movie.addProperty(this.countries, temp.get(i));
            }
        }
        this.model.setNsPrefix("node", this.node_url);
        this.model.setNsPrefix("property", this.properties_uri);
        this.model.setNsPrefix("rdf", this.rdf);
        this.model.setNsPrefix("rdfProperty", this.rdfProperty);
        Writer out = new OutputStreamWriter(new FileOutputStream("output.ttl"),
StandardCharsets.UTF_8);
        this.model.write(out, "TTL");
        out.close();
```

```
    }

    public String createURI(String words) {
        words = words.replaceAll("[^a-zA-Z0-9\\s]", "").toLowerCase();
        String[] Arr = words.split(" ");
        String result = Arr[0];
        if (Arr.length > 1) {
            for (int i=1; i<Arr.length; i++) {
                result += "_" + Arr[i];
            }
        }
        return result;
    }

    public static void main(String[] args) throws IOException {
        BufferedReader Reader = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter NetflixList.csv Relative or Absolute Path: ");
        String filepath = Reader.readLine();
        csvToTriples  q4 = new csvToTriples();
        q4.convertToTTL(filepath);
        System.out.println("Output stored in output.ttl");
    }
}
```

Notes -
Some encoding issue is there while creating TTL file from Jars. This issue doesn't occur while creating TTL directly from code.

Ref -
http://w3schools.sinsixx.com/rdf/rdf_collections.asp.htm
https://www.w3.org/2007/02/turtle/primer/
https://jena.apache.org/documentation/javadoc/jena/org/apache/jena/vocabulary/RDF.html
https://jena.apache.org/documentation/javadoc/jena/org/apache/jena/vocabulary/RDFS.html#subPropertyOf