Problem	1	2	3	Total
Points:	30	30	40	100
Score:				

This homework assignment has 3 problems, for a total of 100 points.

---

1. You are developing a service for helping select Valentine's Day gifts. You need to create a value map from a scale of prices to a scale of user preference. For your target users, gifts that are more expensive are more desirable, but only up to a point (say, $80); gifts that are more expensive than $80 become less and less appropriate.

   - User rankings (best to worst): A, B, C, D, F
   - Prices (in $): 0-20, 20-40, 40-60, 60-80, 80-100, 100-200, 200-500, 500-∞

   (a) (10 points) A mapping meeting the above requirements must be

   A. Total in both directions, but not order preserving in either direction
   B. Total in both directions, and order preserving in each direction
   C. Total and order preserving from rankings to prices but not conversely
   D. Total and order preserving from prices to rankings but not conversely

   (b) (20 points) Devise a pair of mappings from rankings to prices and from prices to rankings that are consistent inverses of each other, or show that such mappings are impossible. Use the style of the figures from the textbook as referenced in the homework assignments. That is, show points and edges, and explain in a few words why your solution satisfies the stated requirement or why no solution can satisfy the stated requirement.

2. Let us model business exception handlers as rules of the form if condition then action. Rules are modeled in OWL. The following trivial rule captures the knowledge that *if the books are missing, then remind and reorder*. This listing presupposes suitable OWL definitions as below.

Listing 1: A trivial rule for handling an exception

```
<Rule rdf:ID="business-exception-1">
 <Condition>
  <Term rdf:ID="missing-books">
    <Test rdf:datatype="&xsd;#string">
    missing
    </Test>
    <onWhat rdf:datatype="&xsd;#string">
    books
    </onWhat>
  </Term>
 </Condition>
 <Action>
  <Sequence rdf:ID="remind+reorder">
   <Step rdf:datatype="&xsd;#string">
   remind
   </Step>
   <Step rdf:datatype="&xsd;#string">
   reorder
   </Step>
  </Sequence>
 </Action>
</Rule>
```

(a) (10 points) In OWL DL, Rule is defined as

A.
```
<owl:Class rdf:about="Rule"/>
```

B.
```
<owl:Class rdf:ID="Rule">
  <rdfs:subClassOf rdf:about='#Condition'/>
</owl:Class>
```

C.
```
<owl:Class rdf:ID="Rule"/>
```

D.
```
<owl:Class rdf:ID="Rule">
  <rdfs:subClassOf rdf:about='#Condition'/>
  <rdfs:subClassOf rdf:about='#Action'/>
</owl:Class>
```

(b) (10 points) In OWL DL, Action is defined as

E.
```
<owl:DatatypeProperty rdf:ID="Action">
  <rdfs:domain rdf:resource="#Rule"/>
  <rdfs:range rdf:resource="#Sequence"/>
</owl:DatatypeProperty>
```

F.
```
<owl:ObjectProperty rdf:ID="Action">
  <rdfs:domain rdf:resource="#Rule"/>
  <rdfs:range rdf:resource="#Sequence"/>
</owl:ObjectProperty>
```

G.
```
<owl:ObjectProperty rdf:ID="Action">
  <rdfs:domain rdf:resource="#Rule"/>
  <rdfs:range rdf:resource="#Step"/>
</owl:ObjectProperty>
```

H.
```
<owl:DatatypeProperty rdf:ID="Action">
  <rdfs:domain rdf:resource="#Rule"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

(c) (10 points) In OWL DL, onWhat is defined as

I.
```
<owl:ObjectProperty rdf:ID="onWhat">
  <rdfs:domain rdf:resource="#Term"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:ObjectProperty>
```

J.
```
<owl:ObjectProperty rdf:ID="onWhat">
  <rdfs:domain rdf:resource="#Term"/>
  <rdfs:range rdf:resource="#Term"/>
</owl:ObjectProperty>
```

K.
```
<owl:Class rdf:ID="onWhat">
  <rdfs:subClassOf rdf:resource="#Term"/>
</owl:Class>
```

L.
```
<owl:DatatypeProperty rdf:ID="onWhat">
  <rdfs:domain rdf:resource="#Term"/>
  <rdfs:range rdf:resource="&xsd;#string"/>
</owl:DatatypeProperty>
```

3. Consider the ontology of Listing 2, which uses the smallerThan and greaterThan properties to enable comparing service providers in terms of some unstated qualities. Based on the usual definition of maximality, we would like to define MaximalProvider as the class of providers that no provider is greater than, and NonMaximalProvider as the class of providers that are not maximal.

Listing 2: A trivial ontology that compares service providers

```
<owl:Class rdf:ID="Comparable"/>
<owl:Class rdf:ID="Provider">
  <rdfs:subClassOf rdf:resource="#Comparable"/>
</owl:Class>

<owl:TransitiveProperty rdf:ID="smallerThan">
  <rdfs:domain rdf:resource="#Comparable"/>
  <rdfs:range rdf:resource="#Comparable"/>
</owl:TransitiveProperty>

<owl:TransitiveProperty rdf:ID="greaterThan">
  <owl:inverseOf rdf:resource="#smallerThan"/>
</owl:TransitiveProperty>
```

(a) (20 points) We can define NonMaximalProvider in OWL DL as

A.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#smallerThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

B.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Provider"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#smallerThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

C.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Provider"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#greaterThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

D.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#smallerThan"/>
      <owl:allValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

E.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <owl:disjointWith>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#greaterThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </owl:disjointWith>
</owl:Class>
```

F.

```
<owl:Class rdf:ID="NonMaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#smallerThan"/>
      <owl:allValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#greaterThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

(b) (20 points) We can define MaximalProvider in OWL DL as

G.

```
<owl:Class rdf:ID="MaximalProvider">
  <owl:disjointWith rdf:resource="#NonMaximalProvider"/>
</owl:Class>
```

H.

```
<owl:Class rdf:ID="MaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#greaterThan"/>
      <owl:allValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

I.

```
<owl:Class rdf:about="Provider">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#MaximalProvider"/>
    <owl:Class rdf:about="#NonMaximalProvider"/>
  </owl:unionOf>
</owl:Class>
```

J.

```
<owl:Class rdf:ID="MaximalProvider">
  <owl:disjointWith rdf:resource="#NonMaximalProvider"/>
</owl:Class>

<owl:Class rdf:about="Provider">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#MaximalProvider"/>
    <owl:Class rdf:about="#NonMaximalProvider"/>
  </owl:unionOf>
</owl:Class>
```

K.

```
<owl:Class rdf:ID="MaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#greaterThan"/>
      <owl:someValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

L.

```
<owl:Class rdf:ID="MaximalProvider">
  <rdfs:subClassOf rdf:resource="#Provider"/>
  <not>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#smallerThan"/>
      <owl:allValuesFrom rdf:resource="#Provider"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  </not>
</owl:Class>

<owl:Class rdf:about="Provider">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#MaximalProvider"/>
    <owl:Class rdf:about="#NonMaximalProvider"/>
  </owl:unionOf>
</owl:Class>
```