CSE 7320 Final Report

Student Name: Chenming Cui
Student ID: 47781790

Part A

- **Codes**

.n3 file code:

```
@prefix CC: <http://www.smu.edu/~47781790/#>.
@prefix schema: <http://schema.org>.

CC:Jimmy a schema:Person.
CC:Jimmy schema:givenName "Jimmy".
CC:Jimmy schema:affiliation CC:Google.
CC:Jimmy schema:knows CC:Marla.

CC:Marla a schema:Person.
CC:Marla schema:givenName "Marla".

CC:Google a schema:Organization.
CC:Google schema:name "Google".
CC:Google schema:location CC:California.

CC:California a schema:Place.
```

SPARQL1.txt

```
prefix CC: <http://www.smu.edu/~47781790/#>
prefix schema: <http://schema.org>

SELECT ?x
WHERE{
    ?x a schema:Person.
}
```

SPARQL2.txt

```
prefix CC: <http://www.smu.edu/~47781790/#>
prefix schema: <http://schema.org>

SELECT ?name ?location
```

```
WHERE{
    ?name a schema:Organization.
    ?name schema:location ?location.
}
```

SPARQL3.txt

```
prefix CC: http://www.smu.edu/~47781790/#
prefix schema: http://schema.org

SELECT ?x
WHERE{
    CC:Jimmy schema:affiliation ?x
}
```

SPARQL4.txt

```
prefix CC: http://www.smu.edu/~47781790/#
prefix schema: http://schema.org

SELECT ?knowsMarla
WHERE{
    CC:Jimmy schema:knows ?x.
    BIND (exists{?x schema:givenName "Marla"} AS ?existsMarla)
    BIND (IF(?existsMarla, "yes", "no") AS ?knowsMarla)
}
```

- **Results**

```
-----------------------------------------------------------------
---  Your Default Namespace Triples                           ---
-----------------------------------------------------------------
@prefix schema: <http://schema.org> .
@prefix CC:    <http://www.smu.edu/~47781790/#> .

CC:California  a  schema:Place .

CC:Marla  a               schema:Person ;
        schema:givenName  "Marla" .

CC:Jimmy  a                 schema:Person ;
        schema:affiliation  CC:Google ;
        schema:givenName    "Jimmy" ;
        schema:knows        CC:Marla .

CC:Google  a              schema:Organization ;
        schema:location   CC:California ;
        schema:name       "Google" .
JerrydeMacBook-Pro:FinalExam JeremyCui$
```

.n3 file capture

```
q : [sparql1.txt]
d : [PartA.n3]
Number of parameters = 2
-----------------------------------------------------------------
---  Your Default Namespace Triples                             ---
-----------------------------------------------------------------
@prefix CC:     <http://www.smu.edu/~47781790/#> .
@prefix schema: <http://schema.org> .

CC:California  a  schema:Place .

CC:Marla  a                schema:Person ;
        schema:givenName   "Marla" .

CC:Jimmy  a                  schema:Person ;
        schema:affiliation   CC:Google ;
        schema:givenName     "Jimmy" ;
        schema:knows         CC:Marla .

CC:Google  a              schema:Organization ;
        schema:location   CC:California ;
        schema:name       "Google" .

SPARQL Query Results:
( ?x = <http://www.smu.edu/~47781790/#Marla> )
( ?x = <http://www.smu.edu/~47781790/#Jimmy> )
```

SPARQL1.txt result

```
q : [sparql2.txt]
d : [PartA.n3]
Number of parameters = 2
-----------------------------------------------------------------
---  Your Default Namespace Triples                             ---
-----------------------------------------------------------------
@prefix CC:     <http://www.smu.edu/~47781790/#> .
@prefix schema: <http://schema.org> .

CC:California  a  schema:Place .

CC:Marla  a                schema:Person ;
        schema:givenName   "Marla" .

CC:Jimmy  a                  schema:Person ;
        schema:affiliation   CC:Google ;
        schema:givenName     "Jimmy" ;
        schema:knows         CC:Marla .

CC:Google  a              schema:Organization ;
        schema:location   CC:California ;
        schema:name       "Google" .

SPARQL Query Results:
( ?name = <http://www.smu.edu/~47781790/#Google> ) ( ?location = <http://www.smu.edu/~47781790/#California> )
```

SPARQL2.txt result

```
q : [sparql3.txt]
d : [PartA.n3]
Number of parameters = 2
-----------------------------------------------------------------
---  Your Default Namespace Triples                           ---
-----------------------------------------------------------------
@prefix CC:     <http://www.smu.edu/~47781790/#> .
@prefix schema: <http://schema.org> .

CC:California  a  schema:Place .

CC:Marla  a                schema:Person ;
        schema:givenName   "Marla" .

CC:Jimmy  a                  schema:Person ;
        schema:affiliation   CC:Google ;
        schema:givenName     "Jimmy" ;
        schema:knows         CC:Marla .

CC:Google  a               schema:Organization ;
        schema:location    CC:California ;
        schema:name        "Google" .

SPARQL Query Results:
( ?x = <http://www.smu.edu/~47781790/#Google> )
```

SPARQL3.txt result

```
q : [sparql4.txt]
d : [PartA.n3]
Number of parameters = 2
-----------------------------------------------------------------
---  Your Default Namespace Triples                           ---
-----------------------------------------------------------------
@prefix CC:     <http://www.smu.edu/~47781790/#> .
@prefix schema: <http://schema.org> .

CC:California  a  schema:Place .

CC:Marla  a                schema:Person ;
        schema:givenName   "Marla" .

CC:Jimmy  a                  schema:Person ;
        schema:affiliation   CC:Google ;
        schema:givenName     "Jimmy" ;
        schema:knows         CC:Marla .

CC:Google  a               schema:Organization ;
        schema:location    CC:California ;
        schema:name        "Google" .

SPARQL Query Results:
( ?knowsMarla = "yes" )
```

SPARQL4.txt result

Part B

- **Codes**

```
import nltk

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

#Create the first sentence
sentence1 = "Jimmy works at Google in California. He was seen
whispering to Marla."

tokens1 = nltk.word_tokenize(sentence1)
tags1 = nltk.pos_tag(tokens1)
tree1 = nltk.ne_chunk(tags1)

print(tree1)

#Create a word list including “works”.
#And define a grammar to choose VP.
work_word_list = ["works", "hired", "contract", "paid", "employed"]
output = "Jimmy"
grammar = "VP: {<VB.*><TO|IN>?<NN.*>}"
cp = nltk.RegexpParser(grammar)

result = cp.parse(tags1)

print(result)

for tuples in result:
    if len(tuples) > 2: #select VP tuples
        if tuples[0][0] in work_word_list: #select target VP
            for couples in tuples: #add the words into the output
                output += " "
                output += couples[0]

print(output)

#Create another sentence to finish the third step
sentence2 = "Jimmy was seen whispering to Marla."
```

```
tokens2 = nltk.word_tokenize(sentence2)
tags2 = nltk.pos_tag(tokens2)
tree2 = nltk.ne_chunk(tags2)

print(tree2)

#Create a word list contains “whispering”
#And create a grammar to find VP
know_word_list = ["knows", "whispering", "contact", "calls", "talks"]
output2 = "Jimmy"
knows = False
grammar2 = "VP: {<VB.*>+<TO|IN>?<NN.*>}"
cp2 = nltk.RegexpParser(grammar2)

result2 = cp2.parse(tags2)

print(result2)

for tuples in result2:
    if len(tuples) > 2: #select VP tuples
        for i in range(len(tuples)):
            if tuples[i][0] in know_word_list: #select target VP
                output2 += " knows " #imply that A knows B
            if i == (len(tuples)-1):
                output2 += tuples[i][0] #find out who is B

print(output2)
```

- **Results**

```
[5]: print(tree1)

(S
  (PERSON Jimmy/NNP)
  works/VBZ
  at/IN
  (ORGANIZATION Google/NNP)
  in/IN
  (GPE California/NNP)
  ./.
  He/PRP
  was/VBD
  seen/VBN
  whispering/VBG
  to/TO
  (PERSON Marla/NNP)
  ./.)
```

Result of tree1

In this result, it shows that Jimmy and Marla belong to PERSON Google belong to ORGANIZATION and California is a GPE, which means that California is a Place.

Thus, I can generate the following N3:

```
CC:Jimmy a schema:Person.
CC:Jimmy schema:givenName "Jimmy".

CC:Google a schema:Organization.
CC:Google schema:name "Google".

CC:California a schema:Place.

CC:Google schema:location CC:California.

CC:Marla a schema:Person.
CC:Marla schema:givenName "Marla".
```

```
[6]: work_word_list = ["works", "hired", "contract", "paid", "employed"]
output = "Jimmy"
grammar = "VP: {<VB.*><TO|IN>?<NN.*>}"
cp = nltk.RegexpParser(grammar)

result = cp.parse(tags1)

print(result)

for tuples in result:
    if len(tuples) > 2:
        if tuples[0][0] in work_word_list:
            for couples in tuples:
                output += " "
                output += couples[0]

print(output)
```

```
(S
  Jimmy/NNP
  (VP works/VBZ at/IN Google/NNP)
  in/IN
  California/NNP
  ./.
  He/PRP
  was/VBD
  seen/VBN
  (VP whispering/VBG to/TO Marla/NNP)
  ./.)
Jimmy works at Google
```

Step 2 Result

From this result, I generated the following N3:

```
CC:Jimmy schema:affiliation CC:Google.
```

```
[9]: know_word_list = ["knows", "whispering", "contact", "calls", "talks"]
output2 = "Jimmy"
knows = False
grammar2 = "VP: {<VB.*>+<TO|IN>?<NN.*>}"
cp2 = nltk.RegexpParser(grammar2)

result2 = cp2.parse(tags2)

print(result2)

for tuples in result2:
    if len(tuples) > 2:
        for i in range(len(tuples)):
            if tuples[i][0] in know_word_list:
                output2 += " knows "
            if i == (len(tuples)-1):
                output2 += tuples[i][0]

print(output2)
```

```
(S
  Jimmy/NNP
  (VP was/VBD seen/VBN whispering/VBG to/TO Marla/NNP)
  ./.)
Jimmy knows Marla
```

Step 3 Result

From this result, I generated the following N3:

```
CC:Jimmy schema:knows CC:Marla.
```

Part C

- **Results**

```
SPARQL Query Results:
( ?x = <http://www.smu.edu/~47781790/#Marla> )
( ?x = <http://www.smu.edu/~47781790/#Jimmy> )
```

SPARQL1.txt result

```
SPARQL Query Results:
( ?name = <http://www.smu.edu/~47781790/#Google> ) ( ?location = <http://www.smu.edu/~47781790/#California> )
```

SPARQL2.txt result

```
SPARQL Query Results:
( ?x = <http://www.smu.edu/~47781790/#Google> )
```

SPARQL3.txt result

```
SPARQL Query Results:
( ?knowsMarla = "yes" )
```

SPARQL4.txt result