Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

Assignment 3

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

Develop the context-free grammar

A. In this problem, you need to develop the context-free grammar that can parse all the following sentences. You will use Python environment to test your grammar to make sure that it does the job and save your Python screenshot.

There are four sentences which we have to parse through our context free grammar. Each senetence is formed of different grammatical structure. In this exercise I will try to explain the structure of each sentence in detail to make it easy for the reader to under the step by step approach. I have folowed recursive syntactic structure, the parser will go from left to right looking for the word. I have used a number of abbrevations for this question in my grammer. It stands as:

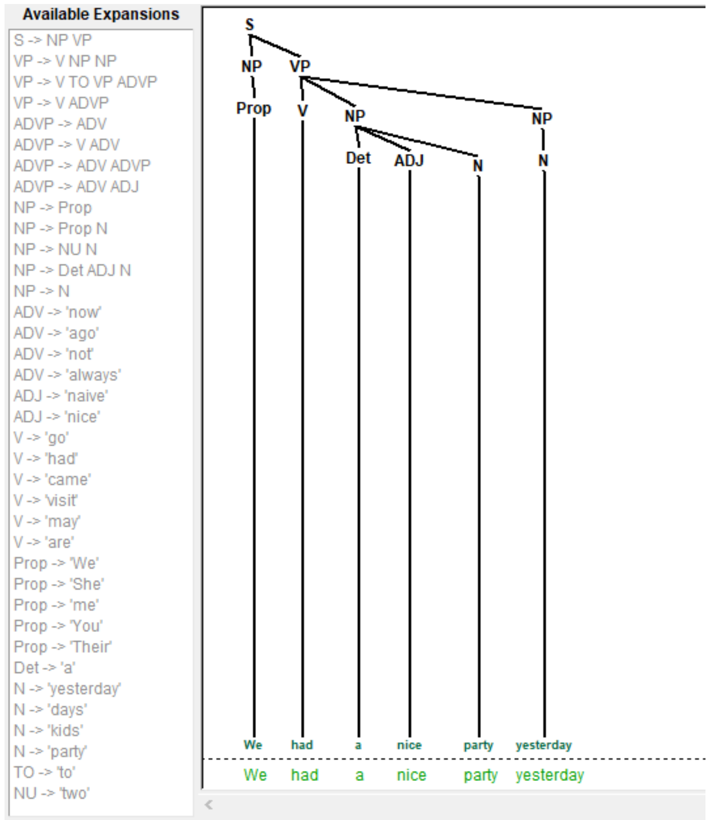
S: Sentence
NP: Noun Phrase
VP: Verb Phrase
Prop: Proper Noun
N: Noun
V: Verb
Det: Determinant
ADVP: Adverb Phrase
ADJ: Adjective

1. We had a nice party yesterday

S	We had a nice party yesterday
NP	Prop
VP	V NP NP
NP	Det ADJ N
NP	N
Prop	We
V	had
Det	a
ADJ	nice
N	party
N	yesterday

In this specific sentence, the sentence is formed of NP (Noun Phrase) and a VP (Verb Phrase). Furthermore, VP has a V (verb) and two NP; NP has two parts one is Det (Determinant) ADJ (Adjective) and N (Noun), the other one has just a N. The verbs and noun has specific words under them, which are as follows: Prop-> We, V-> had, Det-> a, ADJ-> nice, N-> party, yesterday. To show it more intuitively, I also implemented it on the rdparser app to create a tree and see if the grammar fits the sentence.

Name: Raghav Raheja
Course: IST664
SUID: 840334338



As we can see that the grammar perfectly fits the sentence, this can be verified with the output from python:

```
....
(S
  (NP (Prop We))
  (VP (V had) (NP (Det a) (ADJ nice) (N party)) (NP (N
yesterday))))

In [3]:
```

It can be seen that for the specific sentence, there is only one grammar, so there is no ambiguity for the specific sentence. It can be also seen that the sentence has correctly parsed through the grammar through top down parsing in the app and also through the implementation in python.

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

2. She came to visit me two days ago

S	She came to visit me two days a
NP	Prop
VP	V TO VP ADVP
VP	V NP NP
NP	NU N
ADVP	ADV
Prop	She
Prop	me
V	came
TO	to
V	visit
NU	two
N	days
ADV	ago

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun), while the VP has two parts, one has V (Verb) TO (to) VP ADVP (Adverb Phrase) and the other one has V NP NP. Furthermore ADVP has ADV (Adverb) and NP has NU (Number) N (Noun). The terminals words that are in these Non-terminal words are as follows: Prop -> She, me, V -> came, visit, TO -> to, NU -> two, ADV - > ago.

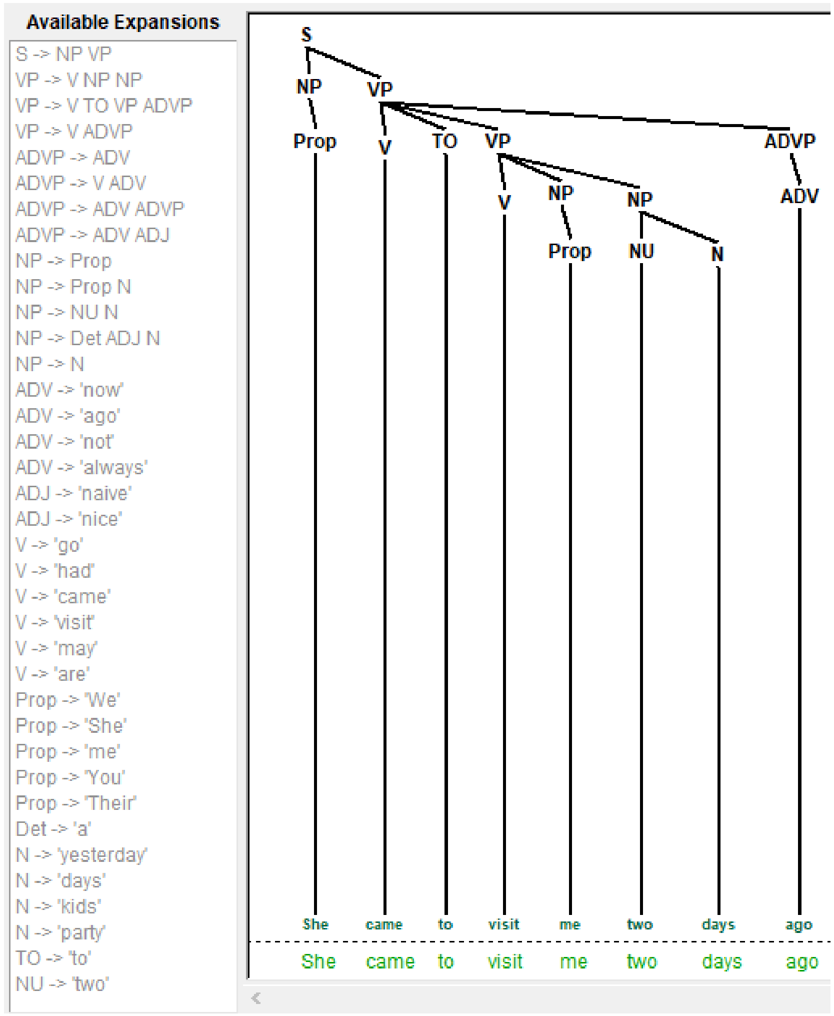
Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

```
(S
  (NP (Prop She))
  (VP
    (V came)
    (TO to)
    (VP (V visit) (NP (Prop me)) (NP (NU two) (N days)))
    (ADVP (ADV ago))))
```

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

Name: Raghav Raheja
Course: IST664
SUID: 840334338

3. You may go now

S	You may go now
NP	Prop
VP	V ADVP
ADVP	V ADV
Prop	You
V	may
V	go
ADV	now

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun), while the VP has V (Verb) ADVP (Adverb Phrase) and ADVP has V ADV (Adverb). The terminals words that are covered for this specific sentence are as follows: Prop -> You, V -> may, go, ADV - > ago.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

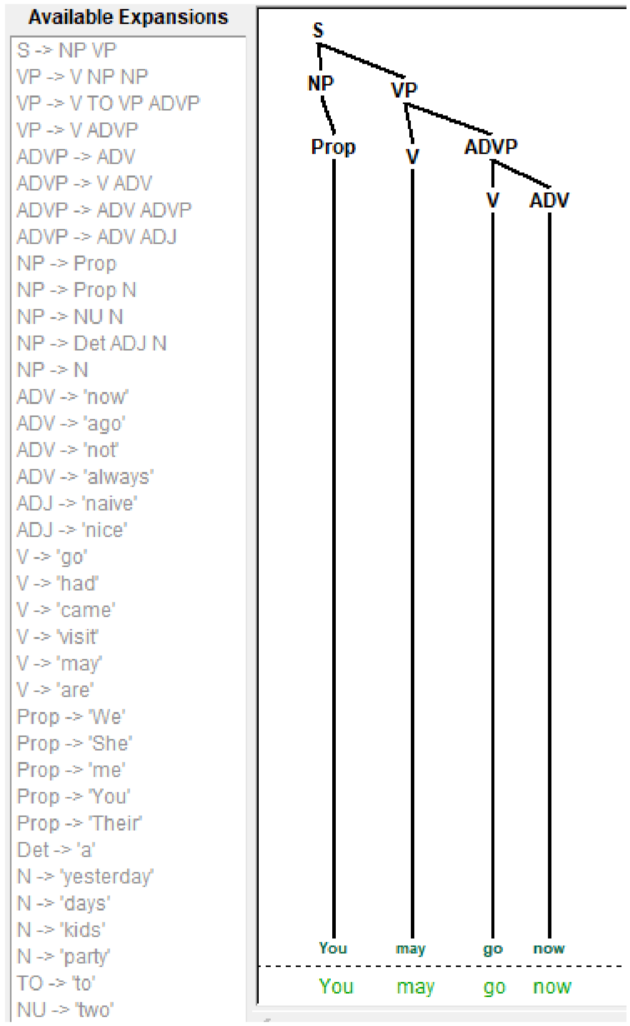
```
    ...:
(S (NP (Prop You)) (VP (V may) (ADVP (V go) (ADV now))))

In [5]:
```

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

4. Their kids are not always naive

S	Their kids are not always naive
NP	Prop N
VP	V ADVP
ADVP	ADV ADVP
ADVP	ADV ADJ
N	kids
V	are
ADV	not
ADV	always
ADJ	naïve

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun) N (Noun), while the VP has V (Verb) ADVP (Adverb Phrase) and ADVP has two parts, ADV (Adverb) ADVP and ADV ADJ (Adjective). The terminals words that are covered from this specific sentence are as follows: Prop -> Their, V -> are, N -> kids, ADJ -> naive, ADV - > not, always.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

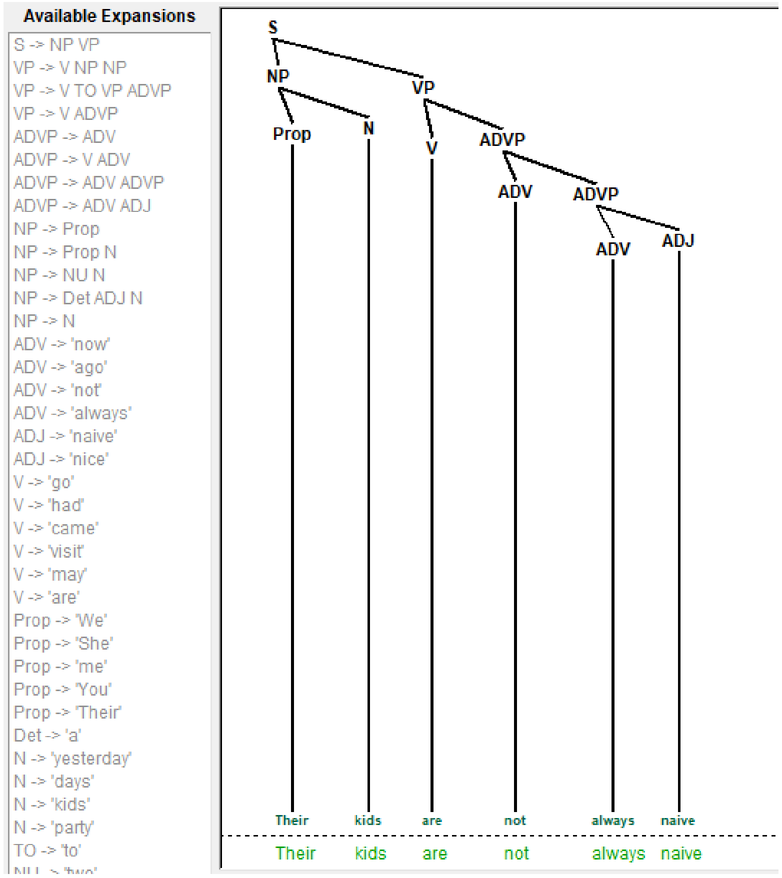
```
   ....
(S
  (NP (Prop Their) (N kids))
  (VP (V are) (ADVP (ADV not) (ADVP (ADV always) (ADJ naive)))))

In [6]:
```

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

Assignment 3

<u>Prof. Lu Xiao</u>

Name: Raghav Raheja
Course: IST664
SUID: 840334338

Additional Sentences:
To work more on the grammar that we created for the four sentences used above, we will be taking four sentence to see if the grammar works on them.

a. Their kids had a nice party yesterday

S	Their kids had a nice party yesterday
NP	Prop N
VP	V NP NP
NP	Det ADJ N
NP	N
Prop	Their
V	had
Det	a
ADJ	nice
N	party
N	yesterday
N	kids

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun) N (Noun), while the VP has V (Verb) NP NP and NP has two parts, Det (Determinant) ADJ (Adjective) N (Noun) and N. The terminals words that are covered from this specific sentence are as follows: Prop -> Their, V -> had, N -> kids, party, yesterday, Det -> a.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

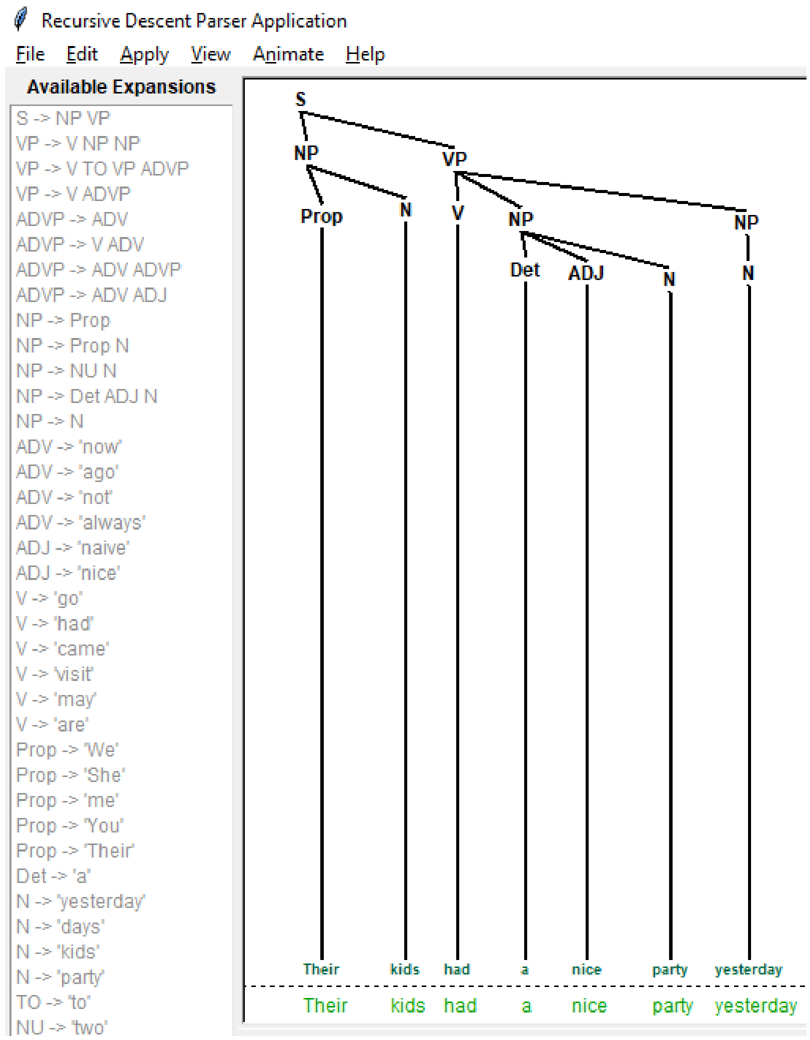
```
   ....
(S
  (NP (Prop Their) (N kids))
  (VP (V had) (NP (Det a) (ADJ nice) (N party)) (NP (N yesterday))))

In [7]:
```

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

Assignment 3
Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

b. You are not always nice

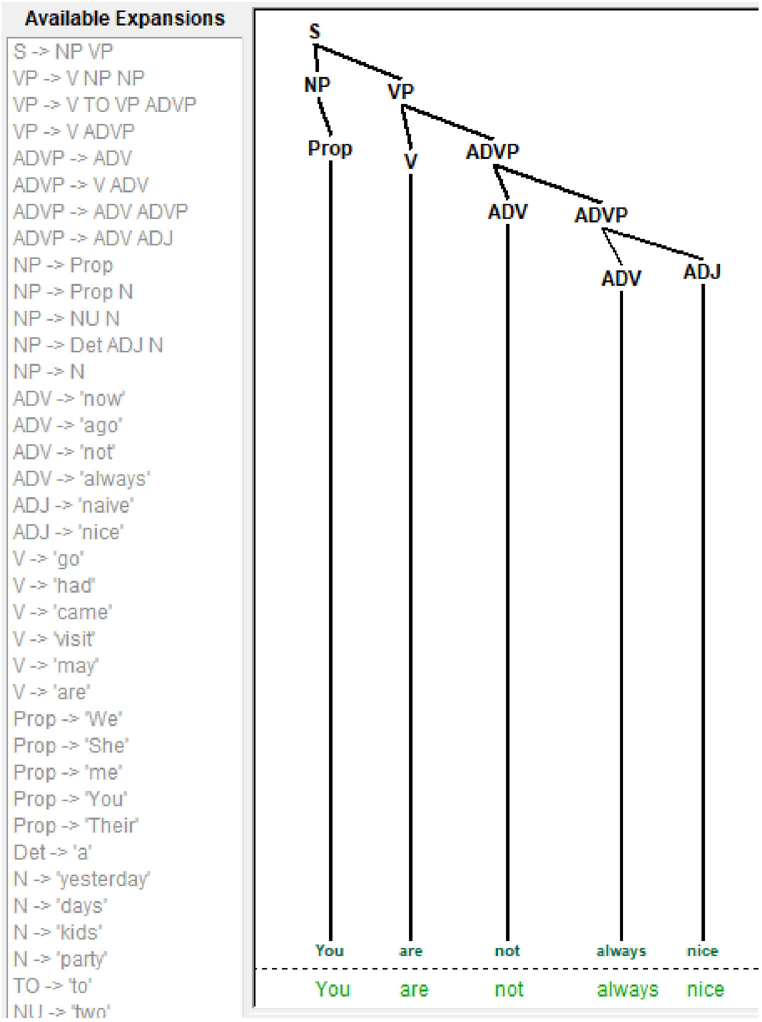
S	You are not always nice
NP	Prop
VP	V ADVP
ADVP	ADV ADVP
ADVP	ADV ADJ
Prop	You
V	are
ADV	not
ADV	always
ADJ	nice

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun), while the VP has V (Verb) ADVP (Adverb Phrase) and ADVP has two parts, ADV (Adverb) ADVP and ADV ADJ (Adjective). The terminals words that are covered from this specific sentence are as follows: Prop -> You, V -> are, ADV -> not, always, ADJ -> nice.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

```
  ....
(S
  (NP (Prop You))
  (VP (V are) (ADVP (ADV not) (ADVP (ADV always) (ADJ nice)))))
```

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

c. Their kids may go now

S	Their kids may go now
NP	Prop N
VP	V ADVP
ADVP	V ADJ
Prop	Their
V	may
V	go
ADJ	now
N	kids

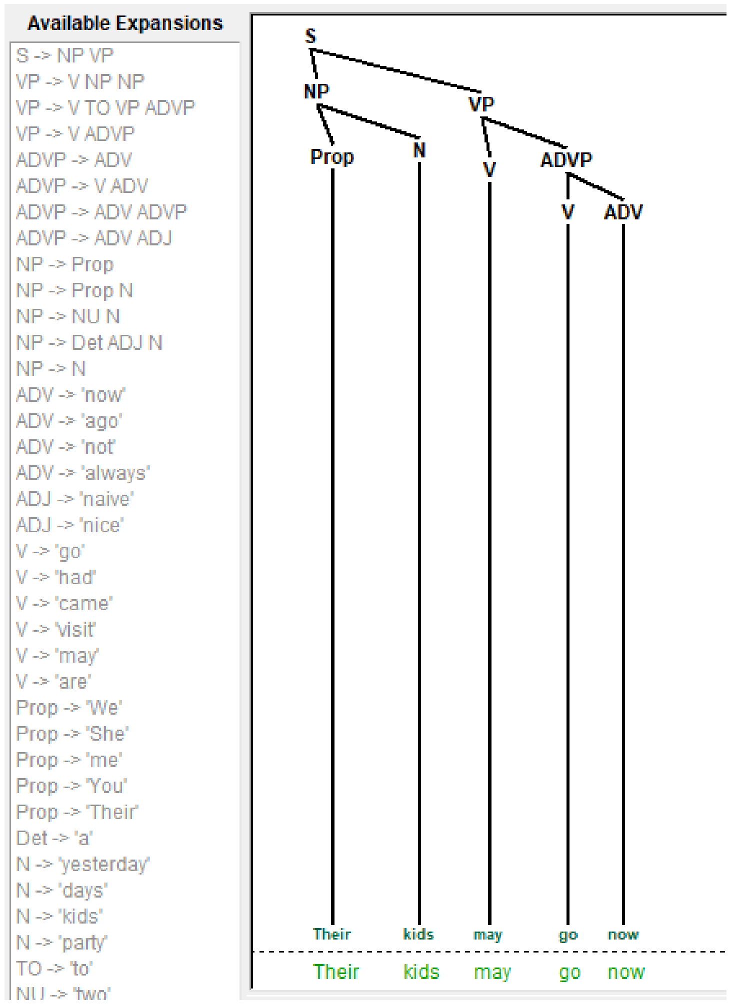
This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun) N (Noun), while the VP has V (Verb) ADVP (Adverb Phrase) and ADVP has V (Verb) ADJ (Adjective). The terminals words that are covered from this specific sentence are as follows: Prop -> Their, V -> may, go, ADJ -> now, N -> kids.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

```
(S (NP (Prop Their) (N kids)) (VP (V may) (ADVP (V go) (ADV now))))

In [9]:
```

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. We can see from the above results that there is one and only one tree for the sentence in our grammar, therefore we can say that there is no ambiguity for our sentence.

Name: Raghav Raheja
Course: IST664
SUID: 840334338

d. Their party had kids yesterday

S	Their party had kids yesterday
NP	Prop N
VP	V NP NP
NP	N
Prop	Their
V	had
N	kids
N	yesterday
N	party

This sentence has two parts, NP (Noun Phrase) and VP (Verb Phrase). NP has a Prop (Proper Noun) N (Noun), while the VP has V (Verb) NP NP and NP has N. The terminals words that are covered from this specific sentence are as follows: Prop -> Their, V -> had, N -> party, kids, yesterday.

Looking at the grammar and parsing structure above, we can see that thee exists only one grammar that can parse the specific sentence. There should not be ambiguity looking at the grammar. This will be seen by looking at the output in python, which we will see below. There is also an implementation of the sentence in the top down parser in the app to confirm that the sentence goes through the parser. The result can be seen below:

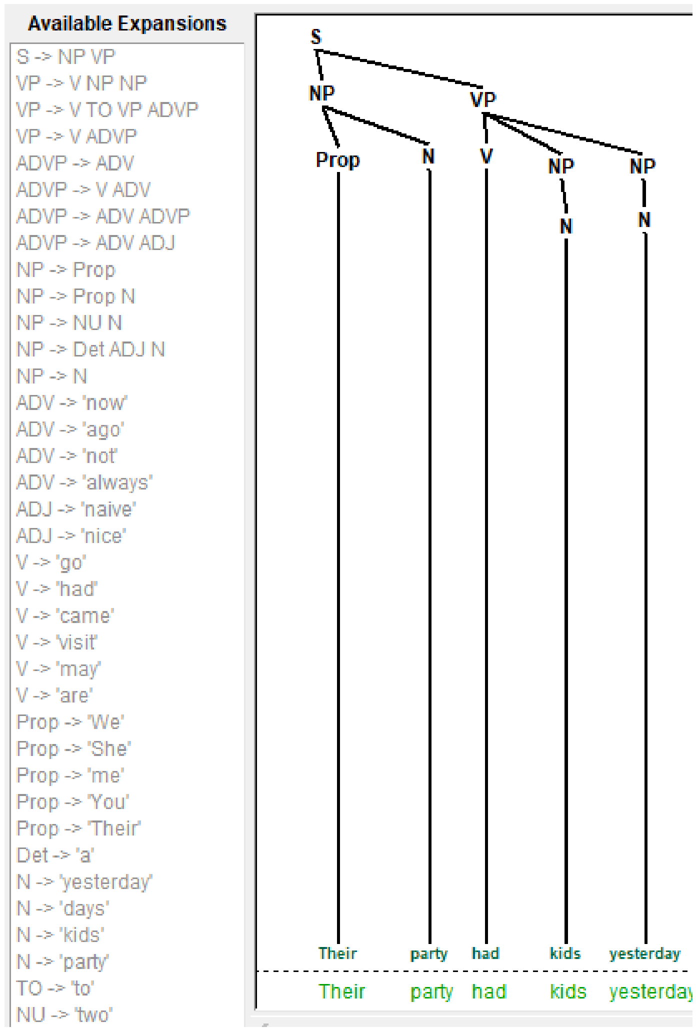
```
(S
  (NP (Prop Their) (N party))
  (VP (V had) (NP (N kids)) (NP (N yesterday))))

In [10]:
```

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338



We can see that our grammar allows the parsing of this sentence and it covers all the terminals and non-terminals used in the sentence. If we look at the grammar and the tree from top down parser, we can see that there exists only one tree, so it is not ubiquitous. However, if we closely look at the sentence, it can be interpreted in two ways: one as, party (political party) had kids and the other one as party (event) had kids (kids were present). This kind of ambiguity is because of the lexicon definition. Although, the grammar will remain the same, but in terms of the meaning of the sentence, it will depend on the context of the conversation to which meaning it is referring.

The questions asks us to generate one sentence that does not make sense. I have interpreted this question as a sentence that has two different trees or is an ambiguous sentence. It is very difficult to create a sentence that has two structural trees that will pass through the grammar, this is because, the grammar has been made for a very small set of corpus, and each path has specific direction, so it becomes highly unlikely for a sentences to go in two different directions, since each type of a terminal symbol is covered

Name: Raghav Raheja
Course: IST664
SUID: 840334338

under just one Non- Terminal. For example, party can be a noun (Social gathering, political group) as well as a verb (enjoy oneself). In our grammar, we have considered party as just a noun and not a verb, so it reduces the probability of ambiguity. Henceforth, it becomes really difficult to generate an ambiguous sentences since each terminal belongs to one and only one non terminal. The other reason is that we have created the grammar just keeping in mind four sentences and that reduces the options of traversing through different grammars.

Probabilistic context-free grammar

B. Now, assuming that the above four sentences are your mini-mini training corpus, you will write a probabilistic context-free grammar. You will use Python environments to test your grammar and save the Python screenshot.

We combine all the above grammars to form one combined grammar which will look like:

```
S -------> NP VP
VP -----> V NP NP | V TO VP ADVP | V ADVP
ADVP -> ADV | V ADV | ADV ADVP | ADV ADJ
NP -----> Prop | Prop N | NU N | Det ADJ N | N
ADV ---> "now" | "ago" | "not" | "always"
ADJ ----> "naive" | "nice"
V -------> "go" | "had" | "came" | "visit" | "may" | "are"
Prop ---> "We" | "She" | "me" | "You" | "Their"
Det ----> "a"
N ------> "yesterday" | "days" | "kids" | "party"
TO -----> "to"
NU ----> "two"
```

There are a number of ways to create the grammar. I could have used more branches under VP and NP however I tried to use minimum branches wherever I could. I considered a sentence as a subject and a predicate, where I considered a subject as a Noun Phrase and the predicate as the Verb Phrase. The Noun Phrase can either be a statement stating something about the noun or it can be a noun in itself. On the other hand, the Verb Phrase states the interrelation of the verb with other grammars. In our corpus, the words or terminal symbols are limited, so we have a limited size of the grammar and there is no need to add extra Non-Terminals. In my grammar, I have used Adverb Phrase (Non terminal) instead of just putting in the adverbs (terminal symbols) as well as there were a couple of sentences where adverb is followed by the adjective or followed by another adverb. However, I did not create a Non-Terminal symbol for Adjective Phrase as it was covered in one way or the other in other Non- Terminals symbols. I just created a terminals symbol that had the words. In the next step, I calculated the occurrences of all the terminals and non-terminals to calculate the probability of the sentences from the grammar. The exact calculation can be shown as below:

S ----------------->	NP VP					
# of occurrences	4					
Percentage	1					
VP --------------->	**V NP NP**	**V TO VP ADVP**	**V ADVP**			
# of occurrences	1	1	2			
Percentage	0.25	0.25	0.5			
ADVP ------------>	**ADV**	**V ADV**	**ADV ADVP**	**ADV ADJ**		
# of occurrences	1	1	1	1		
Percentage	0.25	0.25	0.25	0.25		
NP --------------->	**Prop**	**Prop N**	**NU N**	**Det ADJ N**	**N**	
# of occurrences	3	1	1	1	1	
Percentage	0.43	0.14	0.14	0.14	0.14	
ADV ------------->	**now**	**ago**	**not**	**always**		
# of occurrences	1	1	1	1		
Percentage	0.25	0.25	0.25	0.25		
ADJ -------------->	**naïve**	**nice**				
# of occurrences	1	1				
Percentage	0.5	0.5				
V ----------------->	**go**	**had**	**came**	**visit**	**may**	**are**
# of occurrences	1	1	1	1	1	1
Percentage	0.17	0.17	0.17	0.17	0.17	0.17
Prop ------------->	**We**	**She**	**me**	**You**	**Their**	
# of occurrences	1	1	1	1	1	
Percentage	0.2	0.2	0.2	0.2	0.2	
Det -------------->	**a**					
# of occurrences	1					
Percentage	1					
N ---------------->	**yesterday**	**days**	**kids**	**party**		
# of occurrences	1	1	1	1		
Percentage	0.25	0.25	0.25	0.25		
TO --------------->	**to**					
# of occurrences	1					
Percentage	1					
NU -------------->	**two**					
# of occurrences	1					
Percentage	1					

I have created a list of probabilities considering the four sentences as the corpus. I used Viterbi parser which is a part of NLTK to calculate the probabilities of each sentence:

P_parser = nltk.ViterbiParser(P_assignment_grammer)

The result for each Sentence can be shown as below:

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

1. We had a nice party yesterday

```
(S
  (NP (Prop We))
  (VP
    (V had)
    (NP (Det a) (ADJ nice) (N party))
    (NP (N yesterday)))) (p=2.26478e-06)

In [11]:
```

2. She came to visit me two days ago

```
    ....
(S
  (NP (Prop She))
  (VP
    (V came)
    (TO to)
    (VP (V visit) (NP (Prop me)) (NP (NU two) (N days)))
    (ADVP (ADV ago)))) (p=2.83614e-08)

In [12]:
```

3. You may go now

```
    ....
    ...:
(S (NP (Prop You)) (VP (V may) (ADVP (V go) (ADV now)))) (p=7.40568e-05)

In [13]:
```

4. Their kids are not always naïve

Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

```
(S
  (NP (Prop Their) (N kids))
  (VP
    (V are)
    (ADVP (ADV not) (ADVP (ADV always) (ADJ naive))))) (p=1.15503e-06)

In [14]:
```

Additional Sentences:

a. Their kids had a nice party yesterday

```
    ....
(S
  (NP (Prop Their) (N kids))
  (VP
    (V had)
    (NP (Det a) (ADJ nice) (N party))
    (NP (N yesterday)))) (p=1.87635e-07)
```

b. You are not always nice

```
    ...:
(S
  (NP (Prop You))
  (VP
    (V are)
    (ADVP (ADV not) (ADVP (ADV always) (ADJ nice))))) (p=1.39414e-05)
```

c. Their kids may go now

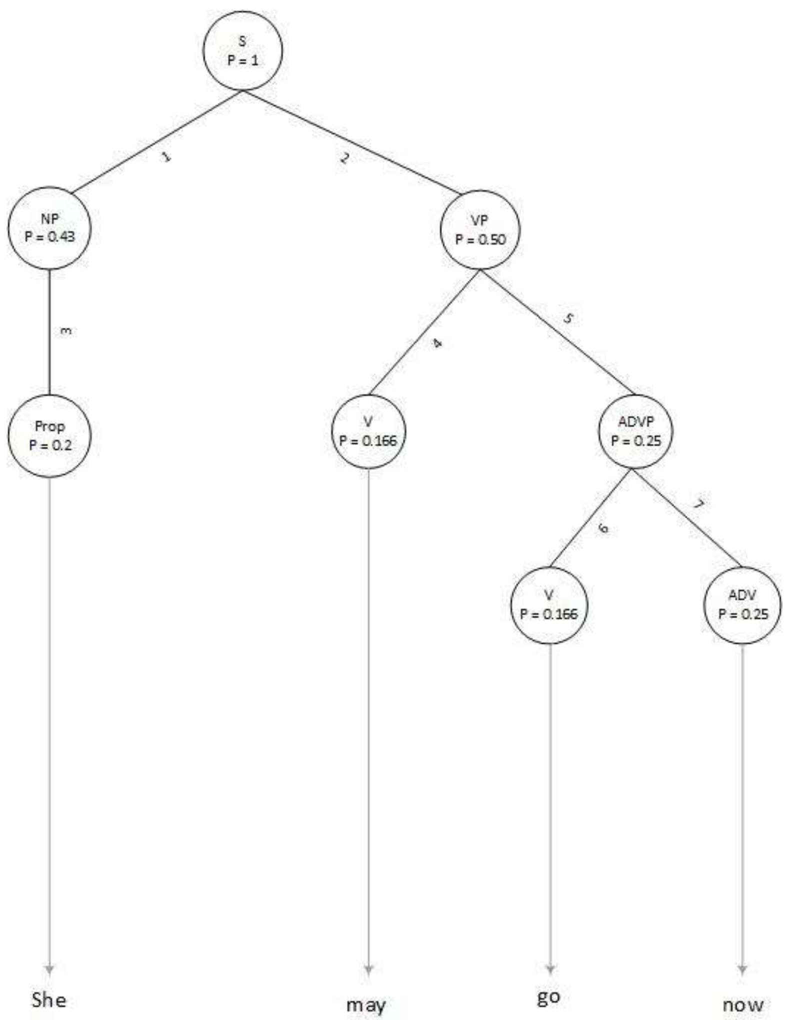
```
    ...:
(S
  (NP (Prop Their) (N kids))
  (VP (V may) (ADVP (V go) (ADV now)))) (p=6.13552e-06)
```

Name: Raghav Raheja
Course: IST664
SUID: 840334338

d. Their party had kids yesterday

```
(S
  (NP (Prop Their) (N party))
  (VP (V had) (NP (N kids)) (NP (N yesterday)))) (p=3.75269e-07)
```

As we can see that the sentence “You may go now” has the highest probability, while the sentence “She came to visit me two days ago” has the lowest probability. We can explain this by the number of branches in the sentence, from my observation, I feel that the sentence will have a lower probability if there are more branches. This can be corroborated by the depiction below:

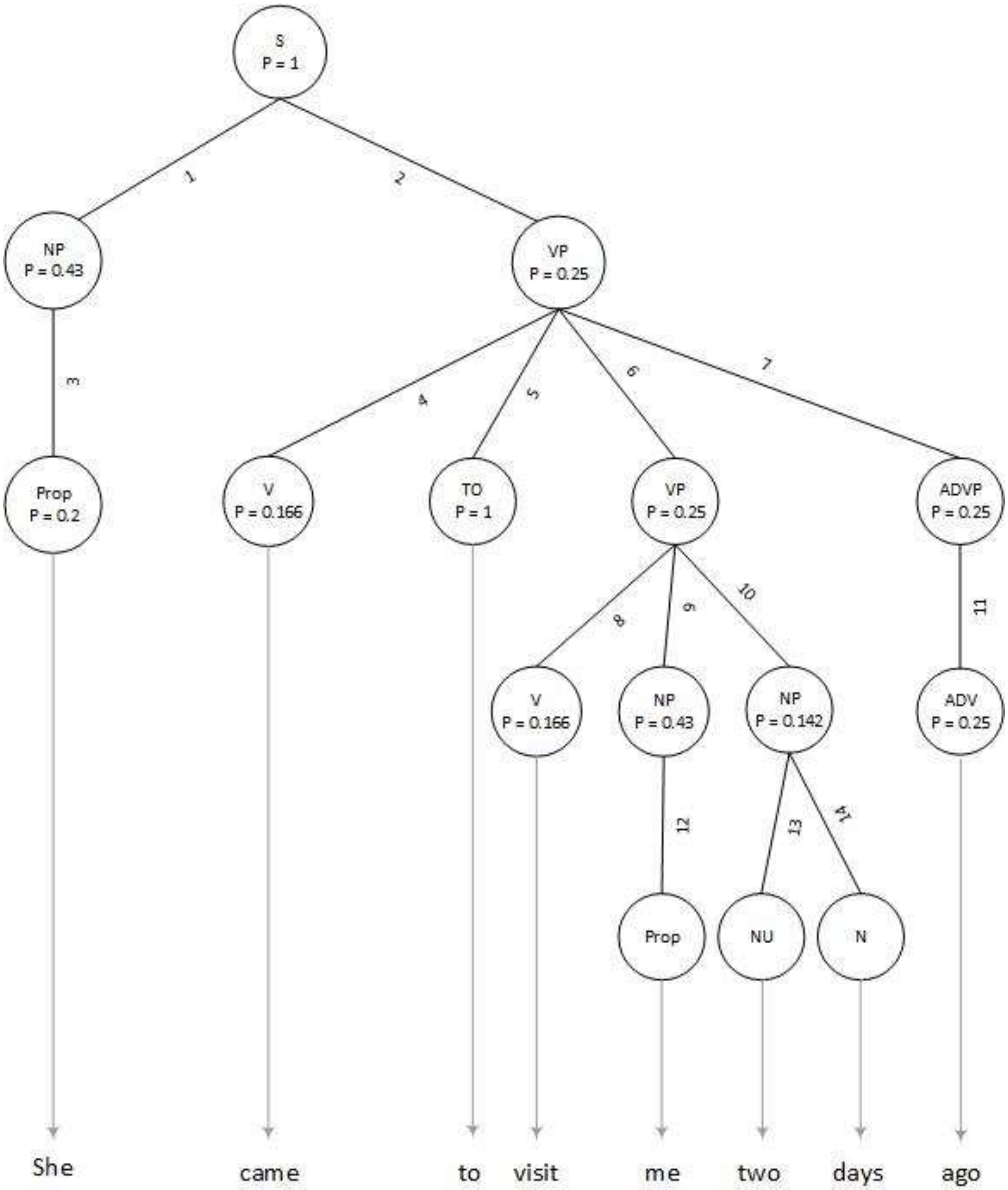


Assignment 3

Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

If we see the tree above, we can see that there are 7 branches for the specific sentence through our grammar and if we get the probability from this tree, it comes same as the calculated one from the Viterbi parser (0.00007405678), which is the highest of all the sentences. On the other hand, if we see for the sentence which has the lowest probability below:



Assignment 3
Prof. Lu Xiao

Name: Raghav Raheja
Course: IST664
SUID: 840334338

We can see from this one that there are 14 branches for this specific sentence through our grammar. It proves our point that more the branches means the probability will keep on reducing. This can be put in other words as the sentence becomes more complex, the branches will increase and hence the probability of the sentence will become less and less. To further get more proof on our observation, I took a sentence randomly with probability lying between the highest and lowest to check if the number of branches actually explains it, and we can see from the tree below that the tree has 10 branches and it makes sense that it lies between the highest and lowest probability.