Report:
The overall output of running my codes is

```
wangyiting@192 python % python3 Proj1.py
Hello Peter.
Peter, the temperature in Irvine now is 68.
Peter, the temperature in Pasadena yesterday is 46.
Peter, the temperature in Tustin yesterday is 56.
dear Peter, now is hotter than yesterday in Irvine
dear Peter, tomorrow is hotter than yesterday in Pasadena
wangyiting@192 python %
```

Question2:The CYK-Parse in the textbook (Fig. 23.5) and in the code require that the syntax be in Chomsky Normal Form. Sometimes this is inconvenient, and it would be helpful to also accept syntax rules of the form *X ⟶ Y [p]* — that is, with a single category on the right hand side. For instance, one of the rules in Fig. 23.4 has such a form. Let's call the modified CNF, relaxed to also permit single category right hand sides, "Chomsky 171 Form". Your assignment in this step is to modify the CYKParse implementation to accept C1F. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the code you modified.

Report:

I change the code so that it can take both the Chomsky Normal Form and CIF form. Since C1F has three parameters and CNF has four parameters, I change the getGrammarSyntaxRules function so that it can return different parameters according to different inputs. Like screenshot below.

```
def getGrammarSyntaxRules(grammar):
    rulelist = []
    for rule in grammar['syntax']:
        if(len(rule)==4):
            yield rule[0], rule[1], rule[2], rule[3]
        else:
            yield rule[0], rule[1], '', rule[2]
```

Alao I make change to CYKParse function so that it can deal with the C1F form text. Like screenshot below.

```
                printV('i:', i, 'j:', j, 'k:', k, '', X, '->', Y, '['+str(p)+']',
                       'PY =' ,getP(Y, i, j), p, '=', getP(Y, i, j) * p)
                PYZ = getP(Y, i, j) * p
                if PYZ > getP(X, i, k):
```

```
                    printV('     inserting from', i, '-
', k, ' ', X, '->', T[Y+'/'+str(i)+'/'+str(j)],
                                'because', PYZ, '=', getP(Y, i, j), '*', p, '>',
getP(X, i, k), '=',
                                'getP(' + X + ',' + str(i) + ',' + str(k) + ')')
                    P[X + '/' + str(i) + '/' + str(k)] = PYZ
                    T[X + '/' + str(i) + '/' + str(k)] = Tree.Tree(X, T[Y+'/'+str
(i)+'/'+str(j)], None)
```

Question3:.Now that your code accepts C1F, take advantage of it. Modify the getGrammarWeather function to use C1F rules~~, with no change in functionality~~. This should make the new grammar somewhat more compact. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the revised getGrammarWeather function.

I create another function getGrammarWeatherC1F() which is based on getGrammarWeather() function but with a little change for my own convenience so that I will not lose the original getGrammarWeather() function data. To make the getGrammarWeatherC1F() function more compact, I add few more rules and delete one rule , ['VP', 'Verb', 'NP', 0.1], which I think is a little bit redundant. And the tests show that the change I make can work well.

And here is my code:

```
def getGrammarWeatherC1F():
    return {
        'syntax' : [
            ['S', 'Greeting', 'S', 0.25],
            ['S', 'NP','VP', 0.25],
            ['S', 'Pronoun', 'VP', 0.25],
            ['S', 'WQuestion', 'VP', 0.25],
            ['VP', 'Verb', 'NP', 0.4],
            ['VP', 'Verb', 'Name', 0.2],
            #['VP', 'Verb', 'NP', 0.1],
            ['VP','Verb','Adjective',0.06],
            ['VP', 'Adverb', 'VP', 0.6 * 0.15],
            ['VP', 'Verb', 'NP+AdverbPhrase', 0.3],
            ['NP', 'Article', 'Noun', 0.5],
            ['NP', 'Adjective', 'Noun', 0.5],
            ['NP','Adjective','AdverbPhrase',0.8],
            ['NP+AdverbPhrase', 'NP', 'AdverbPhrase', 0.2],
            ['NP+AdverbPhrase', 'Noun', 'AdverbPhrase', 0.2],
            ['NP+AdverbPhrase', 'Noun', 'Adverb', 0.2],
            ['NP+AdverbPhrase', 'NP', 'Adverb', 0.15],
            ['NP+AdverbPhrase', 'AdverbPhrase', 'NP', 0.05],
```

```
            ['NP+AdverbPhrase', 'AdverbPhrase', 'Noun', 0.05],
            ['NP+AdverbPhrase', 'Adverb', 'Noun', 0.05],
            ['NP+AdverbPhrase', 'Adverb', 'NP+AdverbPhrase', 0.05],
            ['NP+AdverbPhrase', 'Adverb', 'NP', 0.05],
            ['AdverbPhrase', 'Preposition', 'NP+AdverbPhrase', 0.2],
            ['AdverbPhrase', 'Preposition', 'NP', 0.2],
            ['AdverbPhrase', 'Preposition', 'Name', 0.2],
            ['AdverbPhrase', 'Preposition', 'AdverbPhrase', 0.2],
            ['AdverbPhrase', 'Adverb', 'AdverbPhrase', 0.2],
            ['AdverbPhrase', 'AdverbPhrase', 'Adverb', 0.2],
        ],
        'lexicon' : [
            ['Greeting', 'hi', 0.5],
            ['Greeting', 'hello', 0.5],
            ['WQuestion', 'what', 0.5],
            ['WQuestion', 'will', 0.5],
            ['WQuestion', 'when', 0.25],
            ['WQuestion', 'which', 0.25],
            ['Verb', 'am', 0.5],
            ['Verb', 'is', 0.5],
            ['Verb', 'be', 0.5],
            ['Verb', 'was', 0.5],
            ['Name', 'Peter', 0.1],
            ['Name', 'Sue', 0.1],
            ['Name', 'Irvine', 0.8],
            ['Name', 'Tustin', 0.8],
            ['Name', 'Pasadena', 0.8],
            ['Pronoun', 'I', 1.0],
            ['Noun', 'man', 0.2],
            ['Noun', 'name', 0.2],
            ['Noun', 'temperature', 0.6],
            ['Article', 'the', 0.7],
            ['Article', 'a', 0.3],
            ['Adjective', 'my', 1.0],
            ['Adjective', 'hotter', 0.5],
            ['Adverb', 'now', 0.4],
            ['Adverb', 'yesterday', 0.3],
            ['Adverb', 'today', 0.3],
            ['Adverb', 'tomorrow', 0.3],
            ['Preposition', 'with', 0.5],
            ['Preposition', 'than', 1],
            ['Preposition', 'in', 0.5],
         ]
    }
```

Question4:Here's a bold claim: "The probabilities associated with CNF and C1F lexicons, as seen in Fig. 23.3, are completely unnecessary and have no effect on the output of CYK-Parse." Is this statement true or false? Support your position, ideally by reporting on an experiment with your code. Report: Do you agree with the claim, or not? Back up your position, ideally by reporting on an experiment you ran.

No, I believe that the probability will affect the output somehow although it is not happened in my program now. For instance, if a word has more than one kind of speech, than its kind of speech will affect the outcome.

Question5Improve and extend the weather-bot in several ways:

1. Include the cities Tustin and Pasadena, and the time yesterday, in the system's capabilities. Report: Describe your approach to this step, and an overview of the changes you made. Provide a listing of the revised getGrammarWeather function.

    a.I change the getGrammarWeather function, and add three more parameters to lexicon : ['Name', 'Tustin', 0.8],

    ['Name', 'Pasadena', 0.8],

    ['Adverb', 'yesterday', 0.3],

    b.I change the getTemperature function, to add the location of specific time. Like the screenshot below.

```
    elif location == 'Tustin':
        if time == 'now':
            return '58'
        elif time == 'today':
            return '58'
        elif time == 'tomorrow':
            return '60'
        elif time == 'yesterday':
            return '56'
        else:
            return 'unknown'
    elif location == 'Pasadena':
        if time == 'now':
            return '48'
        elif time == 'today':
```

```
            return '48'
        elif time == 'tomorrow':
            return '50'
        elif time == 'yesterday':
            return '46'
        else:
            return 'unknown'
```

2. Extend the grammar to understand and respond to questions of the sort "Will tomorrow be hotter than today in Irvine", "Will yesterday be hotter than tomorrow in Pasadena?" (Yes, the verb tense doesn't quite work in that one, but you can ignore that.) Report: Describe your approach to this step, and an overview of the changes you made. Provide listings of the code you changed or created.
    a. I change the getGrammarWeather function by adding words and grammar rules to support two questions and analyze them.
    b. To get two time, I use the code below:

```
2.          if leaf[0] == 'WQuestion' and leaf[1] == 'will':
3.              lookingForTime1 = True
4.          if lookingForTime1 and leaf[0] == 'Adverb':
5.              requestInfo['time1'] = leaf[1]
6.              lookingForTime1 = False
7.              continue
8.          if leaf[0] == 'Preposition' and leaf[1] == 'than':
9.              lookingForTime2 = True
10.         if lookingForTime2 and leaf[0] == 'Adverb':
11.             requestInfo['time2'] = leaf[1]
12.             lookingForTime2 = False
```

c.To get the comparable words(like hotter), I use the code

```
if leaf[0] == 'Adjective':
            requestInfo['compare'] = leaf[1]
```

d.After getting time and location, to get the temperature and then answer questions after comparing temperatures, I add the codes below.

```
if requestInfo['time1'] != '':
        if requestInfo['compare'] == 'hotter':
            temp1 = getTemperature(requestInfo['location'],requestInfo['time1'])
            temp2 = getTemperature(requestInfo['location'],requestInfo['time2'])
```

```
            salutation = ''
            if requestInfo['name'] != '':
                salutation = 'dear ' + requestInfo['name'] + ', '
            if temp1 > temp2 :
                salutation += requestInfo['time1'] + ' is hotter than ' + request
Info['time2'] + ' in ' + requestInfo['location']
                print(salutation)
                return
            elif temp1 < temp2:
                salutation += requestInfo['time2'] + ' is hotter than ' + request
Info['time1'] + ' in ' + requestInfo['location']
                print(salutation)
                return
            elif temp2 == temp1:
                salutation += requestInfo['time2'] + ' and ' + requestInfo['time1
'] + ' have the same temperature in ' + requestInfo['location']
                print(salutation)
```

e. The final step is add some tests to main function, like the code below.

```
    T, P = CYKParse.CYKParse(['what', 'was', 'the', 'temperature', 'in', '
Pasadena', 'yesterday'], CYKParse.getGrammarWeatherC1F())
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

    T, P = CYKParse.CYKParse(['what', 'was', 'the', 'temperature', 'in', '
Tustin', 'yesterday'], CYKParse.getGrammarWeatherC1F())
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

    T, P = CYKParse.CYKParse(['will', 'yesterday', 'be', 'hotter', 'than',
 'now', 'in', 'Irvine'], CYKParse.getGrammarWeatherC1F())
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

    T, P = CYKParse.CYKParse(['will', 'yesterday', 'be', 'hotter', 'than',
 'tomorrow', 'in', 'Pasadena'], CYKParse.getGrammarWeatherC1F())
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()
```