Task 1.

What the CFG accepts and rejects:

My grammar rejects all the invalid instances mentioned in the handout, namely my CFG rejects:

- *Je mangent le poisson
- *Les noirs chats mangent le poisson
- *La poisson mangent les chats
- *Je mange les

My CFG accepts:

- Tu regardes la television
- Le chat ne mange pas le poisson
- Je regarde la television

I'll take a bottom up approach to prove that my CFG correctly works on the valid and invalid examples.
Firstly I'll show how my grammar rejects the invalid example "Je mange les". The CFG correctly identifies:

- 'je' as PR-1-Sing
- 'mange' as either V-1-Sing, V-3-Sing-M, or V-3-Sing-F
- 'les' as DT-Plu or PR-do

However, there is no way that 'mange' and 'les' could be connected to create a verb phrase.

As another invalid example, my CFG rejects 'Les noirs chats mangent le poisson'. The CFG correctly identifies:

- 'noris' as A-1-Plu-M
- 'chats' as N-Plu-M

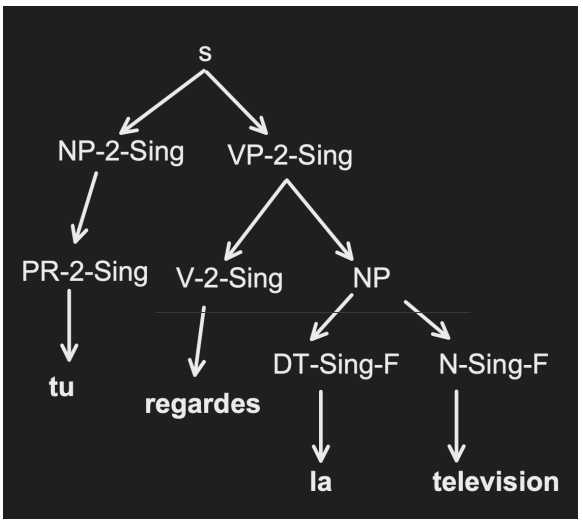
However, since in our grammar the adjective comes after the noun (N-Plu-M A-1-Plu-M) the order of "noris chats" will not be accepted and the sentence will be rejected even though the gender and number of the adjective and noun match.

I'll now go through examples that my CFG accepts and parses.
Take "Tu regardes la television", the CFG correctly identifies:

- 'Tu' as PR-2-Sing
- 'regardes' as V-2-Sing
- 'television' as N-Sing-F

Now, the sentence will be in order "PR-2-Sing V-2-Sing DT-Sing-F N-Sing-F".

- "PR-2-Sing" or "Tu" will create the noun phrase "NP-2-Sing"
- "DT-Sing-F N-Sing-F" or "la television" will create a "NP-3-Sing-F" which can be regarded as an NP
- "V-2-Sing NP" or "regardes la television" will create a "VP-2-Sing"
- And Finally "PR-2-Sing VP-2-Sing" complete the sentence.



Overgeneration and undergeneration:

An example of overgeneration in my CFG is “je mange je”. The grammar will assign “PR-1-Sing” to both “je” and “V-1-Sing” to “mange”. “Je” will make the NP and “mange je " will make the VP. so it’ll pass the grammar even though the sentence doesn’t make sense.

An example of undergeneration is “tu regardes le beau chat mange le poisson noir” as in “you watch the beautiful cat eat the black fish”. Eventhough the CFG identifies “tu regardes le beau chat” and “le beau chat mange le poisson noir”. It can’t manage to deal with the larger sentence constructed from the smaller valid sentences given that the CFG doesn’t have the capability to recognize recursions.

What are some advantages of modeling French grammar with a CFG?
Although I don’t speak French I know that the language is quite complex, with a lot of grammar rules and some irregularities. Given that in CFG, any of the production rules in the grammar can be applied regardless of the context, it would be a great option for modeling a complicated language, like French.
By defining rules at the word level for CFGs, they are capable of modeling a large number of sentences with a small, and compact, number of rules which would be ideal for modeling French.
Furthermore, CFGs are very easy to read, understand, & expand on.

What are some disadvantages of modeling French grammar with a CFG?
Modeling French grammar using CFG can result in slower parsing. This is due to the fact that in CFGs each rule can yield multiple outputs and that in French CFGs we will have many rules. This means that to reach the correct parse the grammar has to iterate through multiple “parse paths” which will result in an explosion of rules. Therefore CFGs might not efficiently scale to the entire French language.

What are some aspects of French grammar that your CFG does not handle?
I don’t speak French, so I might not be able to properly answer this question.

I know that my CFG doesn't account for verbs that are not present tense. Furthermore, I don't consider the indirect objects of verb phrases and multiple adjectives. I'm also not sure that using 'ne' and 'pas' is the only way to make a sentence negative.

Task 2.

To run the file

do :

1.

```
python3 a2-cyk.py -s <"sentance to parse"> -f <path to grammar file>
```

To parse the sentence using the grammar file provided

2.

```
python3 a2-cyk.py -s <"sentance to parse">
```

To parse the sentence with the default grammar file

3.

```
python3 a2-cyk.py
```

To run with the default grammar file and sentence

The default grammar file is "./french-grammar.txt" and the default sentence is "Tu regardes la television"

The output of the file is all possible parses of the sentence both based on the CNF and the original CFG.

ATTENTION: I spend a lot of time figuring out a way to neatly print out the parse tree based on the original CFG. The result is not my best work, given the workload of the end of the semester and multiple other midterms/assignments.
In the end, I resorted to printing the passed tree on the CFG branch by branch, where you have to follow each branch to the end. I build the CFG tree from the CNF tree and use a function to build the connections.
This is an example of "Tu regardes la television". Just start from 's' and go down branch by branch until you reach the terminal nodes, contained in [ ]s. I've also added the CNF based parse tree just in case.

```
This sentance can be produced using the provided CFG.
Possible trees (using):

CNF based Tree:

 [ s  { np-2-sing tu }   [ vp-2-sing  { v-2-sing regardes }   [ np  { dt-sing-f la }   { na-sing-f television }  ]  ]  ]


--------------------------------

CFG based Tree:

s --> np-2-sing
      vp-2-sing



     np-2-sing --> [ pr-2-sing -> 'tu' ]


          vp-2-sing --> v-2-sing
                        np



               v-2-sing --> [ 'regardes' ]


                    np --> np-3-sing-f -> dt-sing-f
                           np-3-sing-f -> na-sing-f



                         dt-sing-f --> [ 'la' ]


                              na-sing-f --> [ n-sing-f -> 'television' ]
```