A student's 66% of code found similar to another student's 70% of code.

.py (66%)	🟥	.py (70%)	🟥
[39-75]()	🟥	[35-69]()	🟥
[6-15]()	🟩	[11-21]()	🟩

The similarities are marked in Red and Green color for convenience.

```
Here goes some comments added by the students
as
program header comment and signature
declarations....................................
...............................................
...............................................
...............................................
```

🟩

```
import re
import pandas as pd

#Creating Analyser class
class Analyser :
    def __init__(self,cleaned_file):
       self.cleaned_file = cleaned_file
       self.len_of_trans = 0
       self.size_of_vocab = 0
       self.num_of_rep = 0



  This Segment contains code segment that
  has no similarity or less similarity ...
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ........
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ........
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ........
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... ..... .....
  ..... ..... ..... ..... ..... .....
```

🟥

```
          return 'Number of pauses made -
indicated by the CHAT symbol (.) are {5}
Statistics {6}' +(str(self.num_of_pauses) +
str(self.statistics_no))


    def analyse_script(self):  # to compute the
statistics and to return all of them combined

        result = ''
        self.tran_len = 0
        f = self.cleaned_file

        for each in f:
```

.py

```
Here goes some comments added by the students as
program header comment and signature
declarations....................................
...............................................
...............................................
...............................................
...............................................
........

...............................................
...............................................
```

🟩

```
import re
import pandas as pd


class Analyser:  # Class created for computing
statistics

    def __init__(self, cleaned_file):
        self.cleaned_file = cleaned_file
        self.tran_len = 0
        self.voc_size = 0
        self.rep_no = 0
   This Segment contains code segment that
   has no similarity or less similarity ...
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... ..... ..... ..
   ... ..... ..... ..... ..... ..... ..... ....
   . ..... ..... ..... ..... ........
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... ..... ..... ..
   ... ..... ..... ..... ..... ..... ..... ....
   . ..... ..... ..... ..... ........
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... .....
   ..... ..... ..... ..... ..... ..... ..... ..
   ... ..... ..... ..... ..... ..... ..... ....
```

🟥

```
                    str(self.pau_no) +
str(self.statistics))

    def analyse_script(self):  # to compute the
statistics and to return all of them combined

        result = ''
        self.tran_len = 0
        in_file = self.cleaned_file
        for element in in_file:
            result += str(element) + " "
        emails = re.findall(r'[.|!|?]', result)
```

```
            result += str(each) + " "
        lt = re.findall(r'[.|!|?]', result)
        lt1 = re.findall(r'\(.*\)', result)
        count1 = len(lt)
        count2 = len(lt1)
        self.len_of_trans = count1 - count2
        self.size_of_vocab =
len(set(re.findall('\w+', result)))
        self.num_of_rep = len(re.findall(r'\
[.\]', result))
        self.retr_words = len(re.findall(r'\
[..\]', result))
        self.gramm_errors =
len(re.findall(r'[*]', result))
        self.num_of_pauses = len(re.findall(r'\
(.\)', result))
        self.statistics_no =
pd.DataFrame({'Length of the transcript':
pd.Series(self.len_of_trans),

'Vocabulary Size':
pd.Series(self.size_of_vocab),
                                        'Number
of Repetition': pd.Series(self.num_of_rep),
                                        'Number
of retractions': pd.Series(self.retr_words),

'Grammatical errors':
pd.Series(self.gramm_errors),
                                        'Number
of Pauses': pd.Series(self.num_of_pauses)})
        return self.statistics_no

    def get_length_transcript(self):  # returns
transcript length alone
        return self.len_of_trans

    def get_vocabulary(self):  # returns
transcript length alone
        return self.size_of_vocab

    def get_repetition(self):  # returns
repetition alone
           return self.num_of_rep
```

```
This Segment contains code segment that
has no similarity or less
similarity ...
..... ..... ..... ..... ..... .....
..... ..... ..... ..... ..... .....
..... ..... ..... ..... ..... ..... ...
.. ..... ..... ..... ..... ..... .....
..... ..... ..... ..... ..... ..... ...
..
```

Note, the green segment is found similar even some careful changing of variable names. Because, the system that we use deployed some some machine learning techniques to check the whole code against all students code and check the syntactical similarity along with structural similarity. It normally ignores a segment of code which is syntactically basic and most of the students have used that. That means, students don't need to be panic about the similarity of very basic code fragments. The system we use for code misuse detection is smart enough to figure that out and ignore them tactfully.

```
        ex1 = re.findall(r'\(.*\)', result)
        count1 = len(emails)
        count2 = len(ex1)
        self.tran_len = count1 - count2
        self.voc_size =
len(set(re.findall('\w+', result)))
        self.rep_no = len(re.findall(r'\[.\]',
result))
        self.ret_no = len(re.findall(r'\[..\]',
result))
        self.gra_no = len(re.findall(r'[*]',
result))
        self.pau_no = len(re.findall(r'\(.\)',
result))
        self.statistics =
pd.DataFrame({'Transcript length':
pd.Series(self.tran_len),

'Vocabulary': pd.Series(self.voc_size),

'Repetition': pd.Series(self.rep_no),

'Retracting': pd.Series(self.ret_no),
                                        'Grammar
Mistakes': pd.Series(self.gra_no),

'Pauses': pd.Series(self.pau_no)})
        return self.statistics

    def get_transcript_length(self):  # returns
transcript length alone
        return self.tran_len

    def get_vocabulary(self):  # returns
transcript length alone
        return self.voc_size

    def get_repetition(self):  # returns
repetition alone
               return self.rep_no
```

```
This Segment contains code segment that
has no similarity or less similarity ...
..... ..... ..... ..... ..... .....
..... ..... ..... ..... ..... .....
..... ..... ..... ..... ..... ..... ....
. ..... ..... ..... ..... ..... ..... ..
... ..... ..... ..... ..... ..... .....
```

The same is true for the red segments. A little change in name is not enough to fool our system. Note in the Red segment, both of them used quite different names. However a careful look will reveal the similarity. This is true for the same task, our code can be almost similar, however, the system that checks for similarity can detect syntax and semantics similarity which are very unusual in case of two students who honestly coded without discussion and without collaboration.