In [ ]:
```
import re
import pandas as pd
```

Read and Store Data

Use f to read txt file.
Store all content in list[0] include newline symbol "\n".

In [ ]:
```
f = open(r"Project1.txt","r")
file = f.read()
list = [file]
```

Get the Upper Line

RE

```
\s+\d+\s\|\s(?P<Player_name>.+)\|
(?P<total_points>[0-9]+[.][0-9])\s+\|
(?P<R1_R>\w)\s+(?P<R1_O>\d{1,})?\|
(?P<R2_R>\w)\s+(?P<R2_O>\d{1,})?\|
(?P<R3_R>\w)\s+(?P<R3_O>\d{1,})?\|
(?P<R4_R>\w)\s+(?P<R4_O>\d{1,})?\|
(?P<R5_R>\w)\s+(?P<R5_O>\d{1,})?\|
(?P<R6_R>\w)\s+(?P<R6_O>\d{1,})?\|
(?P<R7_R>\w)\s+(?P<R7_O>\d{1,})?\|
```

EXAMPLE

```
   28 | SOFIA ADINA STANESCU-BELLU      |3.5  |W  24|D   4|W
22|D  19|L  20|L   8|D  36|
   62 | ASHWIN BALAJI                   |1.0  |W  55|U    |U
|U    |U    |U    |U    |
```

description

We can see in first example the name include a "-". So we use ".+" to match name, it will stop at "|".
In some cases, players did not participate in the competition, so they didn't have an opponent, use "\s+(?P\d{1,})?" to match opponent number, it can match spaces or opponent number.

Output

We will get a list which include dicts

```
[{},{}.....{}]
```

In [ ]:
```
line1 = re.compile("\s+\d+\s\|\s(?P<Player_name>.+)\|(?P<total_points>[0-9]+[.]
line1_result = [m.groupdict() for m in line1.finditer(list[0])]
print("How many players are matched =", len(line1_result))
line1_result
```

what does PXX means?

What does Pxx (eg. P3 or P15) in the rating mean? A P in the rating means "provisional". The USCF gives you a provisional rating until you have played 25 rated games. In layman's terms what this means is that the rating can rise or fall dramatically, until the person has played 25 rated games. In some cases, players who have very high provisional ratings get the rating because they have played other provisional-rated players. Also, a person who has played fewer than 4 rated games is still considered to be unrated, and as such, should be paired as an unrated. This comes into play only in Quads tournaments, where pairings are based on one's rating.

reference:http://weibelchess.blogspot.com/2012/10/some-terminology-for-new-players.html

Get the Lowe Line

RE

```
\s+(?P<state>[A-Z]+)\s\|\s\d+\s\/\sR\:\s
(?P<pre_rating>\d+)(P\d+)?(\s+)?->\d+(P\d+)?
```

EXAMPLE

```
ON | 15771592 / R:  955P11-> 979P18  |     |B    |W    |B    |W
|B    |W    |B    |
MI | 15490981 / R:  377P3 ->1076P10  |     |B    |W    |B    |W
|B    |W    |W    |
MI | 14579262 / R:  967   -> 984     |     |W    |B    |B    |W
|B    |     |     |
```

DESCRIPTION

Use "(?P[A-Z]{2})" to match the state.
Use "(?P\d+)" to match the pre rating.
Whether there are spaces before and after the arrow needs to be judged

In [ ]:
```
line2 = re.compile("\s+(?P<state>[A-Z]{2})\s\|\s\d+\s\/\sR\:\s+(?P<pre_rating>\
line2_result = [m.groupdict() for m in line2.finditer(list[0])]
print("How many players are matched =",len(line2_result))
line2_result
```

```
How many players are matched = 64
```

Out[ ]:

```
[{'state': 'ON', 'pre_rating': '1794'},
 {'state': 'MI', 'pre_rating': '1553'},
 {'state': 'MI', 'pre_rating': '1384'},
 {'state': 'MI', 'pre_rating': '1716'},
 {'state': 'MI', 'pre_rating': '1655'},
 {'state': 'OH', 'pre_rating': '1686'},
 {'state': 'MI', 'pre_rating': '1649'},
 {'state': 'MI', 'pre_rating': '1641'},
 {'state': 'ON', 'pre_rating': '1411'},
 {'state': 'MI', 'pre_rating': '1365'},
 {'state': 'MI', 'pre_rating': '1712'},
 {'state': 'MI', 'pre_rating': '1663'},
 {'state': 'MI', 'pre_rating': '1666'},
 {'state': 'MI', 'pre_rating': '1610'},
 {'state': 'MI', 'pre_rating': '1220'},
 {'state': 'MI', 'pre_rating': '1604'},
 {'state': 'MI', 'pre_rating': '1629'},
 {'state': 'MI', 'pre_rating': '1600'},
 {'state': 'MI', 'pre_rating': '1564'},
 {'state': 'MI', 'pre_rating': '1595'},
 {'state': 'ON', 'pre_rating': '1563'},
 {'state': 'MI', 'pre_rating': '1555'},
 {'state': 'ON', 'pre_rating': '1363'},
 {'state': 'MI', 'pre_rating': '1229'},
 {'state': 'MI', 'pre_rating': '1745'},
 {'state': 'ON', 'pre_rating': '1579'},
 {'state': 'MI', 'pre_rating': '1552'},
 {'state': 'MI', 'pre_rating': '1507'},
 {'state': 'MI', 'pre_rating': '1602'},
 {'state': 'ON', 'pre_rating': '1522'},
 {'state': 'MI', 'pre_rating': '1494'},
 {'state': 'ON', 'pre_rating': '1441'},
 {'state': 'MI', 'pre_rating': '1449'},
 {'state': 'MI', 'pre_rating': '1399'},
 {'state': 'MI', 'pre_rating': '1438'},
 {'state': 'MI', 'pre_rating': '1355'},
 {'state': 'MI', 'pre_rating': '980'},
 {'state': 'MI', 'pre_rating': '1423'},
 {'state': 'MI', 'pre_rating': '1436'},
 {'state': 'MI', 'pre_rating': '1348'},
 {'state': 'MI', 'pre_rating': '1403'},
 {'state': 'MI', 'pre_rating': '1332'},
 {'state': 'MI', 'pre_rating': '1283'},
 {'state': 'MI', 'pre_rating': '1199'},
 {'state': 'MI', 'pre_rating': '1242'},
 {'state': 'MI', 'pre_rating': '377'},
 {'state': 'MI', 'pre_rating': '1362'},
 {'state': 'MI', 'pre_rating': '1382'},
 {'state': 'MI', 'pre_rating': '1291'},
 {'state': 'MI', 'pre_rating': '1056'},
 {'state': 'MI', 'pre_rating': '1011'},
 {'state': 'MI', 'pre_rating': '935'},
 {'state': 'MI', 'pre_rating': '1393'},
 {'state': 'MI', 'pre_rating': '1270'},
 {'state': 'MI', 'pre_rating': '1186'},
 {'state': 'MI', 'pre_rating': '1153'},
 {'state': 'MI', 'pre_rating': '1092'},
 {'state': 'MI', 'pre_rating': '917'},
 {'state': 'MI', 'pre_rating': '853'},
 {'state': 'MI', 'pre_rating': '967'},
```

```
{'state': 'ON', 'pre_rating': '955'},
{'state': 'MI', 'pre_rating': '1530'},
{'state': 'MI', 'pre_rating': '1175'},
{'state': 'MI', 'pre_rating': '1163'}]
```

DESCRIPTION

Convert the list which include dicts to a dataframe object Then merge the two dataframes horizontally

In [ ]:
```
df_result1 = pd.DataFrame(line1_result)
df_result2 = pd.DataFrame(line2_result)
df_mix = pd.concat([df_result1,df_result2,pd.DataFrame(columns=['pre_ave_oppone
print(df_mix)
```

Calculate the average pre-tournament rating

Use the data in dataframe to calculate the average pre-tournament rating

In [ ]:
```
round_opponent_column_index = [3,5,7,9,11,13,15]                              #the co
for i in range(df_mix.shape[0]):                                              #df_mix
    sum_pre_opponents_rating = 0                                              # use i
    count = 0                                                                 # use i
    for p in round_opponent_column_index:                                     # find
        if (df_mix.iloc[i,p] != None):
            sum_pre_opponents_rating += int(df_mix.iloc[int(df_mix.iloc[i,p])-1
            count += 1
    pre_ave_opponents_rating = int(sum_pre_opponents_rating / count)    #calcul
    df_mix.at[i,'pre_ave_opponents_rating'] = pre_ave_opponents_rating  # store
print(df_mix['pre_ave_opponents_rating'])
```

Store the data as csv file

In [ ]:
```
df_mix.to_csv("result.csv")
```