PROJ 201 Project Final Report

Project Title: Tournament design with constraints

Name, Surname & ID of group members: Serhan Yorulmaz 26481, Irmak Çoban 26839, Ömer Faruk Altıntel 26327

Supervised by: Kağan Kurşungöz

Month, day, year: 01/15/2021

SABANCI UNIVERSITY

Abstract

Tournaments are competitive organizations that find the best teams or individuals from certain competitors. Elimination style tournaments are the tournaments that are based on the logic of eliminating less powerful teams or individuals stage by stage in order to obtain a winner at the end. Since every tournament-style depends on different circumstances, their success rates have to be different. In this project, the goal is to test whether the elimination tournament system gives the optimal success rate. With this project, it is investigated which parameters will give the optimal result in determining tournament winners according to variable factors in the actual tournaments. In order to do that, various Python programs that acquire the success rate of systems developed using libraries called “itertools” and “math” and their functions such as “permutations”, “combinations” and “factorial”. At the end of the project, lots of data to evaluate the elimination system with different circumstances were acquired. Results of programs are either explained verbally or given as charts. In conclusion, the elimination system is not a good way of sorting the teams especially for the teams in top-ranks and mid-ranks rather than worst teams. Because the chances of their rankings are very unstable and unpredictable in the elimination system. Also, the best team has a relatively small chance to win the tournament.

Introduction

Tournaments are competitive organizations that find the best teams or individuals from certain competitors. Different tournament types use different parameters for ranking. These parameters are determined by the organizer to find the optimal result. Factors like the number of matches, number of competitors, duration of the tournament, ranking criteria, etc. may differ. Since there are no strict rules about tournament design, the efficiency and optimality of different styles are different (Brown & Minor, n.d.). Even more, some designs have a relatively big risk of giving results that the strongest is not the first. In light of that, optimal tournaments should find the most correct ranking, which is the optimal result, among certain competitors in a minimum time. Optimal results are results in which the success of the winning participants is most accurately evaluated, regardless of the influence of the most reliable and variable factors among other participants in the field of competition (László, 2020). One of the most used tournament designs is the elimination system which is argued in the article of David (1963) in detail with limitations of the system and success rate (pp. 116–127). The purpose of this research is to find out the consistency and the success rate of the elimination system and its feasibility under different circumstances like with different numbers of competitors.

Methods and Materials

In this project, the Python programming language was chosen to create a simulation that can observe the first and second participants of the qualifying tournament system. All codes are used in the Google Colab development environment. The reason for this is that the developers of the project are more familiar with the Python coding language than other languages, and the sample code in this language is more common and accessible on the internet.

The first program that developed gives the output of all possible combinations and possible rankings of these combinations of 4 participant elimination-style tournaments. The aim of this program is to visualize the elimination style tournament and its foreseen results. These results will be lighting our way for tournaments with more participants or with different circumstances. The "itertools" library is imported to obtain the combinations with the combination function. In detail, the output is divided into 4 sections which represent 4 different matching options. The rankings of teams which are "A", "B", "C" and "D" obtained manually with a tree diagram. (appendix1)

The following week's schedule again simulated a tournament with 4 participants. participants were represented as"A", "B"," C "and" D". After these participants are assigned to a list, one of the binary combinations of elements in this list is defined as a list called "route1". For the combination function, the module "itertools" is imported into this program. "route1" is written to the console. In this program, where each participant was eliminated once thanks to a" for loop", the remove function was used for the elimination process, in which the matches in all scenarios and the rest represented the first one. Then, after a participant is eliminated in the first round, one participant is eliminated in the same way among the remaining participants, and after each elimination, one of the binary combinations of the remaining participants is printed on the screen using the combination function. Each binary combination represents a match.

The following week, a 4-participant tournament was simulated again, but this time the participants were assigned numbers to be consecutive and different from each other. These figures represent the strength of the participants. For the program to be easy to write and suitable for this purpose, participants were named with integers 1, 2, 3, and 4. In this way, the indexes of the lists can be compared directly in the comparisons to be made. a list called" matches " was assigned dual permutations of participants. The duos, here again, represent matching and are printed to the console to be well observable. By navigating through all created binaries and comparing their zero and first indexes, a "for loop" is created that will assign the one with the number value greater than them to a new list if it is not already in the list. The integers in the new list represent the participants who qualify for the next route. For this route, binary permutations are taken again and another "for loop" is written in the same logic that makes exactly the same comparisons. another new list that is formed thus represents what remains in the next round, and this process continues until a single binary remains and their large one creates a list alone. When there is only one element left in the list, this element is written to the screen as the first and the last sieve is written to the screen as the second.

However, due to logic errors in this code, the correct result cannot be obtained. Again, a list of "participants" is created with integers 1, 2, 3, and 4 to represent the participants and their strength, and the binary permutations of the elements in this list are written to a list using the permutations function. Another binary that is not in this list is written to a new list so that it corresponds to the index of each binary sublist in this list. A new list of opposing binaries is provided by nested for loops so that this corresponds to all the binaries in the first list. Both lists correspond to each other for a cycle of indexes that are visited and the current index of the list every time these two are compared in binary values the number of elements in the lists, and two lists with one big one in the duo new participant are added to the list. in both lists, the duos corresponding to the current index of the indexes visited in the cycle represent first-round matches. Thus, all possible scenarios were asked to be considered. in both lists, a new list has been added with a number value greater than the binary corresponding to the index currently being processed. Of the elements in this list, the larger one represents the first and the smaller one represents the second. counters are added to the loop to count which participant is the first and second number of times when performing this operation for each encounter scenario. And when the process is complete, these counters are multiplied by 100 after being proportional to twelve, the binary number of permutations of four, which is the percentage of how many times each participant wins and comes second. All this algorithm was applied in exactly the same way for a tournament with 8 participants and 16 participants. For 8 participants, the list of participants consists of integers from one to eight, while for the list of participants of the tournament with 16 participants, it consists of integers from one to sixteen.

A completely new algorithm was established after the previous code traded without considering each scenario and was not effectively applicable to a tournament of 8 and 16 participants.

There are two separate modules imported in the new code, "itertools" and "math". Participants; 'A', ‘B’, 'C', and ' D ' is a 4-participant tournament simulation. All permutations of participants are assigned to a list called "perm". A function is written after it takes this perm list as a parameter. this function is processed as much as the number of elements of the parameter list it receives thanks to the 'while loop'. Of the participants corresponding to the index i and i+1 in each cycle, the stronger one is written into a new list as the participant who is eligible to participate in the next route. And the function returns the list that creates this new route. With this function, the next route is determined until there are two participants left in the new route. and when there are two participants left, this means that the last match will be held. the loser in the last match, that is, the participant who is less than the other as the number value, is determined by the min function and is indicated as the second. This process is done for all permutations of participants through a cycle so that the process is performed for all scenarios. Thanks to the counters placed in this for loop, it is determined how many times each participant is second. By proportioning these counters to the factorial of the length of the participant list and multiplying this ratio by 100, the percentage of each participant being second is calculated.

Using the same algorithm and system, the code can be used to create a simulation of an 8-participant tournament. Only the counter amount is increased until the number of participants is equal, and the results can be observed as a percentage. Although this algorithm could theoretically be applied to a 16-participant tournament, in practice the results could not be observed due to the lack of a computer with equipment capable of performing up to 16 factorial operations at once.

The code written at the next stage aimed to create an 8-participant tournament simulation. In order to simulate situations with this code, such as the possibility of finding participants of equal power value that may be in a real tournament, the participants ' powers were randomly assigned. For this random process, randint is imported from the random module. a dictionary is created by assigning each participant one of the integers from 1 to list length using the randint function. In this way, when a participant is called, it will be possible to process the value assigned to it directly in the program. This time, a new function that works in logic similar to the function written in the previous code is written in the code. this time the function takes two parameters. a parameter is a list of participants, and a parameter is a dictionary. This function takes consecutive two-index elements from the list of participants and also compares them by taking the values assigned to these elements in the dictionary. The comparison process takes place differently this time. At that time, the value of the two elements in the dictionary is summed. A random integer from integers between 1 and this total value is chosen by the randint function. If this random value is between 1 and the value of the first indexed number, it is the winner, but if it is not, the winner is the element corresponding to the other indexed integer value. This function returns a list of participants who have qualified for the next round as a result. As a result of new routes determined using this function, the winner is found as a result of the route remaining single-element, and the second occurs due to the deletion of this winning element from the list of the previous two participants. By processing this process multiple times, the realism of the percentage value obtained can be increased by changing the randomly determined value in the function every time. In the last stage, the percentage of participants being first and second is one hundred times the ratio of each counter to how many times the whole transaction was performed.

For the percentages obtained from the simulation generated by the previous code to approach reality, all this operation must be performed for different permutations of participants (i.e., all possible match scenarios). In the previous code, a list called 'perm' is provided by assigning all permutations of the participant list created with the permutations function; and writing a for loop that will allow it to navigate every version of the permutation immediately inside the for loop indicating how many times we want to perform the operation.

Because all these written programs do not apply to 16-participant tournament simulations, different methods were searched and permutation generators were examined. But the 16-participant simulation could not be written with many failed attempts, due to the lack of coding knowledge of the project participants and the fact that some written programs require practically too long to complete the process.

Last week's code is a different version of the previous one. Since the problem of the previous one is exceeding the RAM memory, this week, RAM friendly code is tried to be written. The logic behind it is keeping only one option in memory, processing it, and deleting it in order to get a new option instead of keeping all the options in memory at the same time. The permutation generator is itertools' permutation function located in the for loop. Thus, every option is evaluated one by one. But this time, another problem occurred. RAM memory was enough to calculate the result but the process would take too much time to see the output. So, we put a counter into the code to calculate the processing time of the code. The counter was calculating 100.000 operations every 0.7 seconds. Since the number of operations is 16!, the whole process would take around 6.63 years. In conclusion, "itertools' permutation" is not suitable for that project.

Results

In the first of the codes mentioned above, the elimination system is based on the elimination of one of the participants each time. A realistic result has not been achieved with this code and the simulation did not give the ratio of the first and second as requested but thanks to the results obtained with this code, the code has been improved. In the code written at the next stage, the binary combinations of participants were eliminated, and these combinations represented the matches that would take place but according to the results obtained, it was observed that this code does not contain all possible scenarios, and with the development of this code, a new code that processes all permutations of participants is processed with all possible permutations. In this way, the scenarios of all possible encounters were taken into account. At this stage, it was observed that the percentage of the strongest participant being first is 100%, the percentage of the second strongest participant being second is 66.6666%, and the percentage of the third strongest participant being second is 33.3333%. As a result of observing the results of this simulation, it was realized that the code was not close enough to reality. There are deficiencies in the algorithm of the simulation that gives these results, such as the strong always defeats the weak. And so the search for a new algorithm was launched. In the newly established algorithm, participants were randomly assigned random numbers from 1 to the number of participants, so that they were closer to the truth, not ignoring the possibility that the weak could defeat the strong. In this code algorithm detailed above, it was observed that each participant has a probability of being the 1st, while even the 2 participants, who are equal to each other and have the lowest power, have a probability of being the first with different percentages such as 0.4861 and 0.489335. While this algorithm is applicable for 8-participant tournaments, it could not be realized for 16-participant tournaments because owned computers do not have the ability to calculate as many odds as 16 factors. The results and outputs obtained can be observed in the appendix section. The output obtained by executing the mentioned code three times is given in the table below.

Outcomes of the first time the code was run:

```
 {'A': 5, 'B': 5, 'C': 2, 'D': 3, 'E': 8, 'F': 2, 'G': 5, 'H': 5}

first place percentages of participants:      second place percentages of participants:

for A: 15.007440476190476                     for A: 14.756696428571429

for B: 14.882192460317462                     for B: 14.876984126984127

for C: 2.6577380952380953                     for C: 6.921626984126984

for D: 5.9759424603174605                      for D: 10.567708333333334

for E: 29.468998015873016                      for E: 15.918650793650793

for F: 2.6634424603174605                      for F: 6.854662698412699

for G: 14.868799603174605                     for G: 14.905009920634921

for H: 14.726190476190476                     for H: 14.947916666666666
```

Outcomes of the second time the code was run:

{'A': 6, 'B': 7, 'C': 3, 'D': 7, 'E': 4, 'F': 5, 'G': 5, 'H': 3}

first place percentages of participants:	second place percentages of participants:
for A: 16.371031746031747	for A: 14.457589285714286
for B: 20.756944444444443	for B: 15.169394841269842
for C: 4.727678571428571	for C: 8.638640873015873
for D: 20.83258928571429	for D: 15.064484126984127
for E: 8.188244047619047	for E: 11.274801587301587
for F: 12.148561507936508	for F: 13.286210317460318
for G: 12.273561507936508	for G: 13.296378968253968
for H: 4.701388888888889	for H: 8.8125

Outcomes of the third time the code was run:

{'A': 7, 'B': 1, 'C': 1, 'D': 2, 'E': 2, 'F': 8, 'G': 7, 'H': 3}

first place percentages of participants:	second place percentages of participants:
for A: 26.445932539682538	for A: 19.411706349206348
for B: 0.7879464285714286	for B: 4.154017857142857
for C: 0.7810019841269842	for C: 4.239583333333333
for D: 3.4754464285714284	for D: 9.789186507936508
for E: 3.3876488095238098	for E: 9.788690476190476
for F: 31.43700396825397	for F: 19.060515873015873
for G: 26.449900793650794	for G: 19.465029761904763
for H: 7.2351190476190474	for H: 14.09126984126984

Outcomes of the fourth time the code was run:

{'A': 6, 'B': 1, 'C': 6, 'D': 8, 'E': 1, 'F': 5, 'G': 6, 'H': 3}

first place percentages of participants:	second place percentages of participants:
for A: 17.794642857142858	for A: 16.743055555555557
for B: 0.4861111111111111	for B: 2.8660714285714284
for C: 17.80902777777778	for C: 16.816964285714285
for D: 26.64831349206349	for D: 17.1812996031746
for E: 0.48933531746031744	for E: 2.9117063492063493
for F: 13.400049603174601	for F: 15.621775793650794
for G: 17.858630952380953	for G: 16.73487103174603
for H: 5.513888888888889	for H: 11.124255952380953

Discussion and Conclusion

When examining the code (appendix 9), which is seen in the above table, it can be observed that the first and second percentages of participants with equal strength are very close to each other. In addition, participants whose strength is equal and the weakest are also very unlikely to be first and second, while their percentages are equal to each other. It can also be observed in the table that the percentages change according to the power distributions in the tournament. Another point observed in the output of codes is the percentage of participants with the largest and second-largest power to be first is greater than the percentage of the second, while the percentage of other participants to be second is greater than the percentage of first. When the results are examined at the specified angles, it can be said that this simulation gives consistent results. But, on the other hand, even the strongest participants of the tournament have a fairly low percentage of first place, which raises doubts about the reliability of the elimination system on the point of determining the first strongest and second strongest. In this tournament system, it can be said that even the strongest participant is only likely to be first with a percentage well below fifty percent, while the lowest participant is likely to be first even with 0.5% percent, this system does not give reliable and stable results enough to find the strongest participants as first and second. In this case, the elimination system can produce extremely unstable results, not only because it is directly proportional to the strength of the winners.

References

Brown, J., & Minor, D. (n.d.). Selecting the Best? Spillover and Shadows in Elimination Tournaments. *Management Science*, *60*(12), 3087-3102.

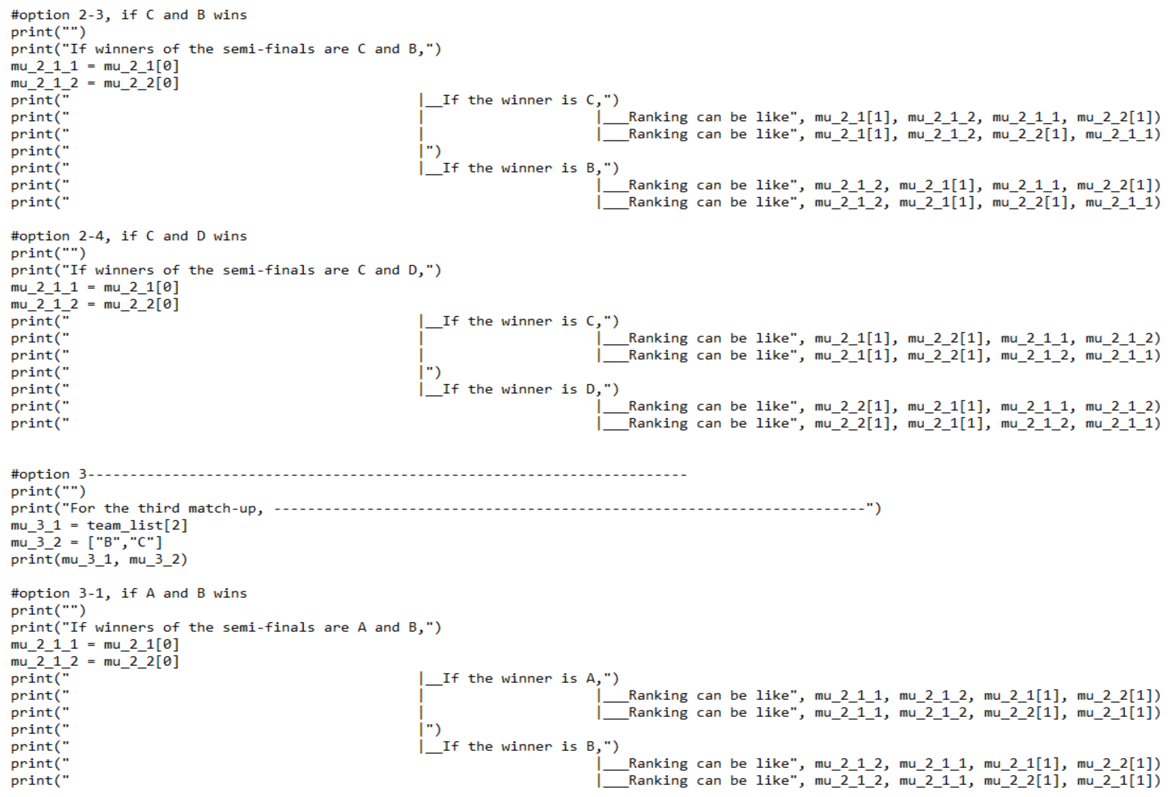
David, H. A. (1963). *The method of paired comparisons* (Vol. 12). London.

László, C. (2020). Optimal Tournament Design: Lessons From the Men's Handball Champions League. *Journal of Sports Economics, 21*(8), 848-868.

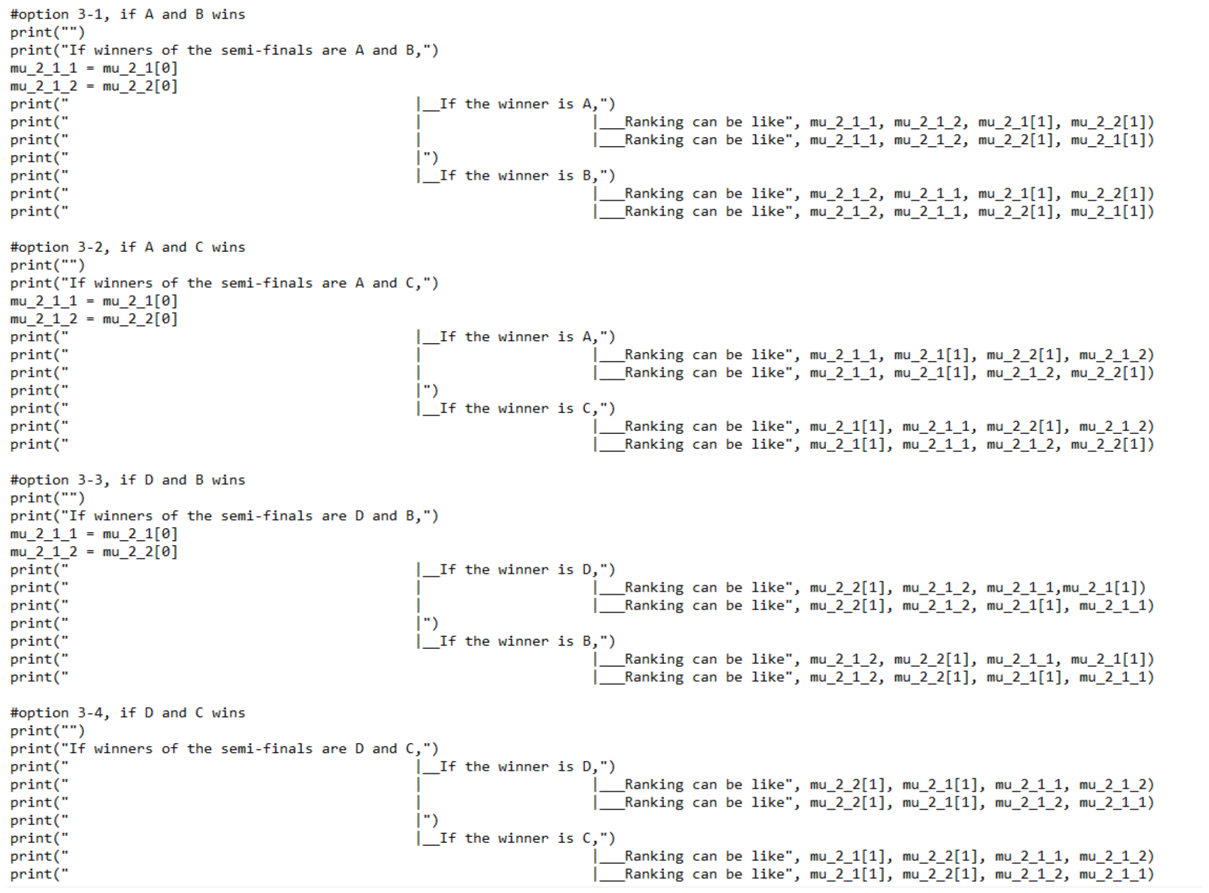
Appendixes

Appendix 1

```
import itertools
team_list = list(itertools.combinations(["A","B","C","D"], 2))

#option 1----------------------------------------------------------------------
print("For the first match-up, ----------------------------------------------------------------------")
mu_1_1 = team_list[0]
mu_1_2 = ["C","D"]
print(mu_1_1, mu_1_2)

#option 1-1, if A and C wins
print("")
print("If winners of the semi-finals are A and C,")
mu_1_1_1 = mu_1_1[0]
mu_1_1_2 = mu_1_2[0]
print("                                          |__If the winner is A,")
print("                                          |                    |___Ranking can be like", mu_1_1_1, mu_1_1_2, mu_1_1[1], mu_1_2[1])
print("                                          |                    |___Ranking can be like", mu_1_1_1, mu_1_1_2, mu_1_2[1], mu_1_1[1])
print("                                          |")
print("                                          |__If the winner is C,")
print("                                                               |___Ranking can be like", mu_1_1_2, mu_1_1_1, mu_1_1[1], mu_1_2[1])
print("                                                               |___Ranking can be like", mu_1_1_2, mu_1_1_1, mu_1_2[1], mu_1_1[1])

#option 1-2, if A and D wins
print("")
print("If winners of the semi-finals are A and D,")
mu_1_2_1 = mu_1_1[0]
mu_1_2_2 = mu_1_2[1]
print("                                          |__If the winner is A,")
print("                                          |                    |___Ranking can be like", mu_1_2_1, mu_1_2_2, mu_1_1[1], mu_1_2[0])
print("                                          |                    |___Ranking can be like", mu_1_2_1, mu_1_2_2, mu_1_2[0], mu_1_1[1])
print("                                          |")
print("                                          |__If the winner is D,")
print("                                                               |___Ranking can be like", mu_1_2_2, mu_1_2_1, mu_1_1[1], mu_1_2[0])
print("                                                               |___Ranking can be like", mu_1_2_2, mu_1_2_1, mu_1_2[0], mu_1_1[1])
#option 1-3, if B and C wins
print("")
print("If winners of the semi-finals are B and C,")
mu_1_3_1 = mu_1_1[1]
mu_1_3_2 = mu_1_2[0]
print("                                          |__If the winner is B,")
print("                                          |                    |___Ranking can be like", mu_1_3_1, mu_1_3_2, mu_1_1[0], mu_1_2[1])
print("                                          |                    |___Ranking can be like", mu_1_3_1, mu_1_3_2, mu_1_2[1], mu_1_1[0])
print("                                          |")
print("                                          |__If the winner is C,")
print("                                                               |___Ranking can be like", mu_1_3_2, mu_1_3_1, mu_1_1[0], mu_1_2[1])
print("                                                               |___Ranking can be like", mu_1_3_2, mu_1_3_1, mu_1_2[1], mu_1_1[0])

#option 1-4, if B and D wins
print("")
print("If winners of the semi-finals are B and D,")
mu_1_4_1 = mu_1_1[1]
mu_1_4_2 = mu_1_2[1]
print("                                          |__If the winner is B,")
print("                                          |                    |___Ranking can be like", mu_1_4_1, mu_1_4_2, mu_1_1[0], mu_1_2[0])
print("                                          |                    |___Ranking can be like", mu_1_4_1, mu_1_4_2, mu_1_2[0], mu_1_1[0])
print("                                          |")
print("                                          |__If the winner is D,")
print("                                                               |___Ranking can be like", mu_1_4_2, mu_1_4_1, mu_1_1[0], mu_1_2[0])
print("                                                               |___Ranking can be like", mu_1_4_2, mu_1_4_1, mu_1_2[0], mu_1_1[0])

#option 2----------------------------------------------------------------------
print("")
print("For the second match-up, ----------------------------------------------------------------------")
mu_2_1 = team_list[1]
mu_2_2 = ["B","D"]
print(mu_2_1, mu_2_2)

#option 2-1, if A and B wins
print("")
print("If winners of the semi-finals are A and B,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                          |__If the winner is A,")
print("                                          |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_1[1], mu_2_2[1])
print("                                          |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_2[1], mu_2_1[1])
print("                                          |")
print("                                          |__If the winner is B,")
print("                                                               |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_1[1], mu_2_2[1])
print("                                                               |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_2[1], mu_2_1[1])

#option 2-2, if A and D wins
print("")
print("If winners of the semi-finals are A and D,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                          |__If the winner is A,")
print("                                          |                    |___Ranking can be like", mu_2_1_1, mu_2_2[1], mu_2_1[1], mu_2_1_2)
print("                                          |                    |___Ranking can be like", mu_2_1_1, mu_2_2[1], mu_2_1_2, mu_2_1[1])
print("                                          |")
print("                                          |__If the winner is D,")
print("                                                               |___Ranking can be like", mu_2_2[1], mu_2_1_1, mu_2_1[1], mu_2_1_2)
print("                                                               |___Ranking can be like", mu_2_2[1], mu_2_1_1, mu_2_1_2, mu_2_1[1])
```



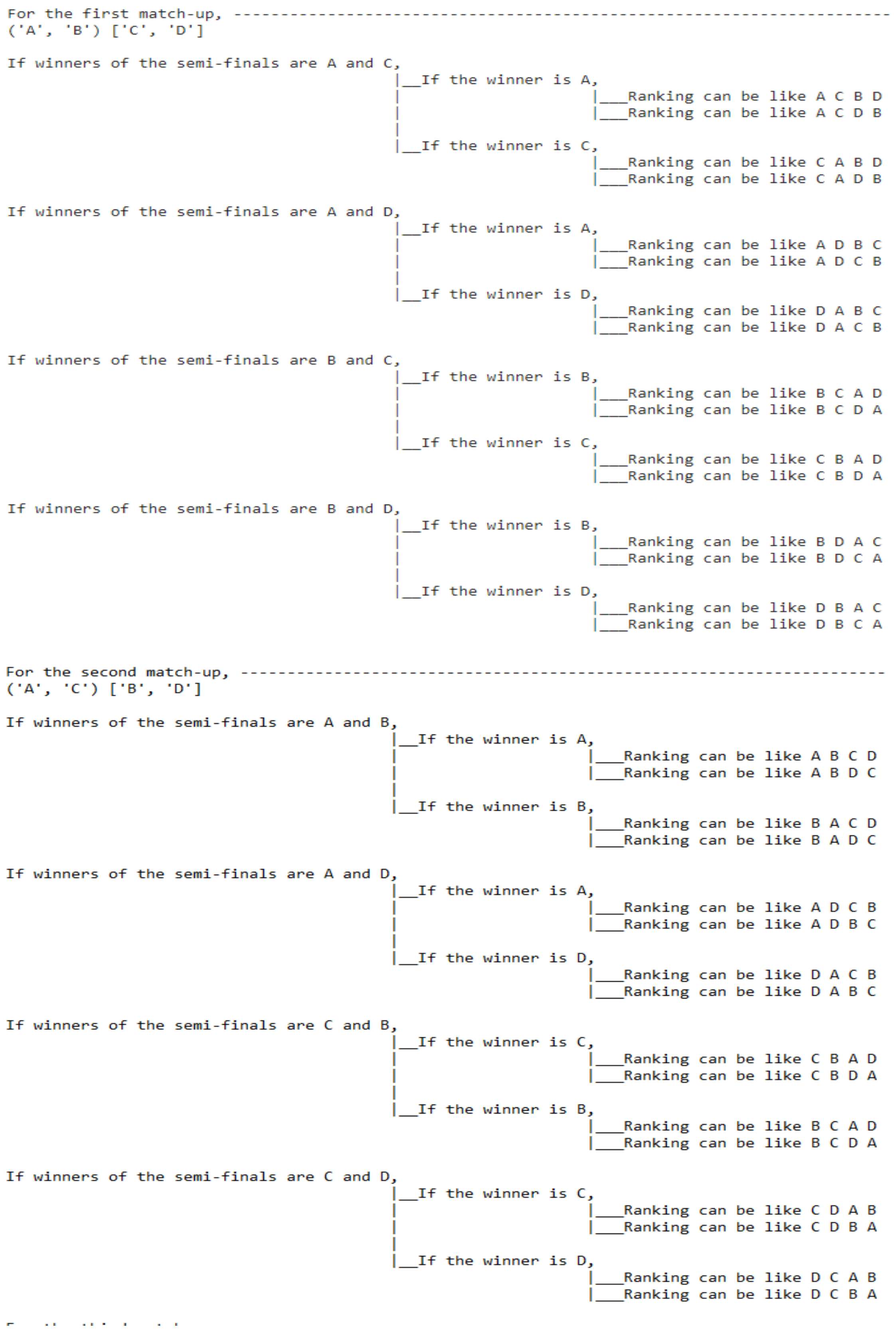
```
#option 2-3, if C and B wins
print("")
print("If winners of the semi-finals are C and B,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is C,")
print("                                         |                    |___Ranking can be like", mu_2_1[1], mu_2_1_2, mu_2_1_1, mu_2_2[1])
print("                                         |                    |___Ranking can be like", mu_2_1[1], mu_2_1_2, mu_2_2[1], mu_2_1_1)
print("                                         |")
print("                                         |__If the winner is B,")
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1[1], mu_2_1_1, mu_2_2[1])
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1[1], mu_2_2[1], mu_2_1_1)

#option 2-4, if C and D wins
print("")
print("If winners of the semi-finals are C and D,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is C,")
print("                                         |                    |___Ranking can be like", mu_2_1[1], mu_2_2[1], mu_2_1_1, mu_2_1_2)
print("                                         |                    |___Ranking can be like", mu_2_1[1], mu_2_2[1], mu_2_1_2, mu_2_1_1)
print("                                         |")
print("                                         |__If the winner is D,")
print("                                                              |___Ranking can be like", mu_2_2[1], mu_2_1[1], mu_2_1_1, mu_2_1_2)
print("                                                              |___Ranking can be like", mu_2_2[1], mu_2_1[1], mu_2_1_2, mu_2_1_1)


#option 3-----------------------------------------------------------------------
print("")
print("For the third match-up, ----------------------------------------------------------------------")
mu_3_1 = team_list[2]
mu_3_2 = ["B","C"]
print(mu_3_1, mu_3_2)

#option 3-1, if A and B wins
print("")
print("If winners of the semi-finals are A and B,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is A,")
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_1[1], mu_2_2[1])
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_2[1], mu_2_1[1])
print("                                         |")
print("                                         |__If the winner is B,")
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_1[1], mu_2_2[1])
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_2[1], mu_2_1[1])
```



```
#option 3-1, if A and B wins
print("")
print("If winners of the semi-finals are A and B,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is A,")
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_1[1], mu_2_2[1])
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1_2, mu_2_2[1], mu_2_1[1])
print("                                         |")
print("                                         |__If the winner is B,")
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_1[1], mu_2_2[1])
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_1_1, mu_2_2[1], mu_2_1[1])
```

```
#option 3-2, if A and C wins
print("")
print("If winners of the semi-finals are A and C,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is A,")
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1[1], mu_2_2[1], mu_2_1_2)
print("                                         |                    |___Ranking can be like", mu_2_1_1, mu_2_1[1], mu_2_1_2, mu_2_2[1])
print("                                         |")
print("                                         |__If the winner is C,")
print("                                                              |___Ranking can be like", mu_2_1[1], mu_2_1_1, mu_2_2[1], mu_2_1_2)
print("                                                              |___Ranking can be like", mu_2_1[1], mu_2_1_1, mu_2_1_2, mu_2_2[1])

#option 3-3, if D and B wins
print("")
print("If winners of the semi-finals are D and B,")
mu_2_1_1 = mu_2_1[0]
mu_2_1_2 = mu_2_2[0]
print("                                         |__If the winner is D,")
print("                                         |                    |___Ranking can be like", mu_2_2[1], mu_2_1_2, mu_2_1_1,mu_2_1[1])
print("                                         |                    |___Ranking can be like", mu_2_2[1], mu_2_1_2, mu_2_1[1], mu_2_1_1)
print("                                         |")
print("                                         |__If the winner is B,")
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_2[1], mu_2_1_1, mu_2_1[1])
print("                                                              |___Ranking can be like", mu_2_1_2, mu_2_2[1], mu_2_1[1], mu_2_1_1)

#option 3-4, if D and C wins
print("")
print("If winners of the semi-finals are D and C,")
print("                                         |__If the winner is D,")
print("                                         |                    |___Ranking can be like", mu_2_2[1], mu_2_1[1], mu_2_1_1, mu_2_1_2)
print("                                         |                    |___Ranking can be like", mu_2_2[1], mu_2_1[1], mu_2_1_2, mu_2_1_1)
print("                                         |")
print("                                         |__If the winner is C,")
print("                                                              |___Ranking can be like", mu_2_1[1], mu_2_2[1], mu_2_1_1, mu_2_1_2)
print("                                                              |___Ranking can be like", mu_2_1[1], mu_2_2[1], mu_2_1_2, mu_2_1_1)
```

Appendix1 Output



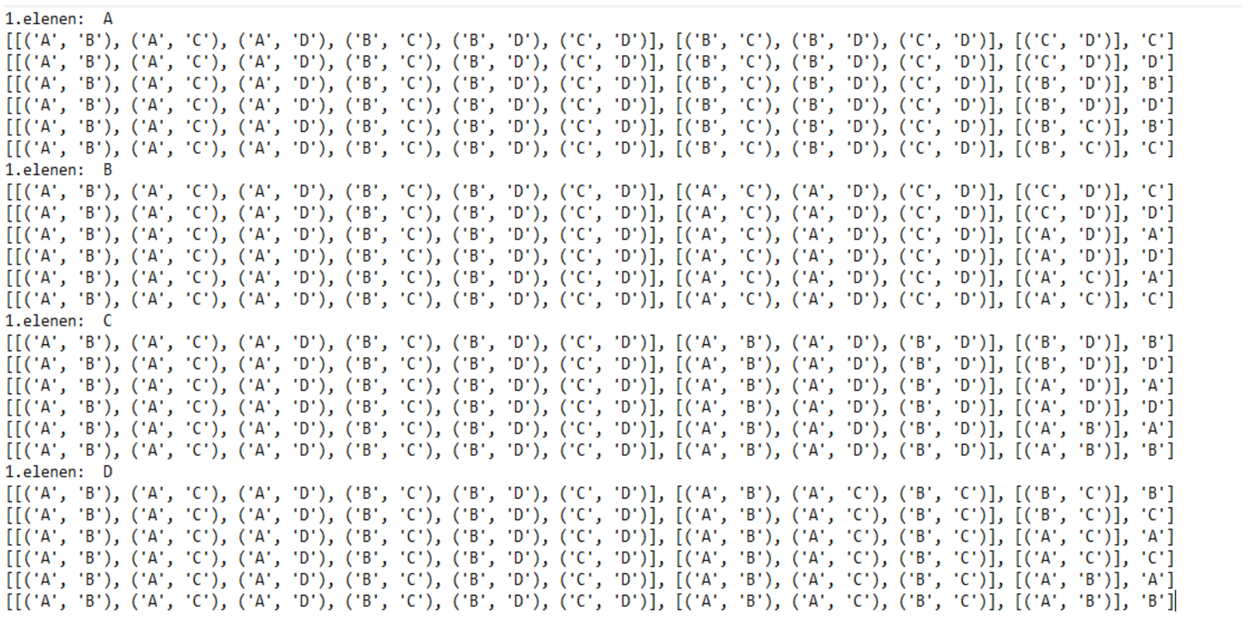
```
For the first match-up, ----------------------------------------------------------------------
('A', 'B') ['C', 'D']

If winners of the semi-finals are A and C,
                                          |__If the winner is A,
                                          |                      |___Ranking can be like A C B D
                                          |                      |___Ranking can be like A C D B
                                          |
                                          |__If the winner is C,
                                                                 |___Ranking can be like C A B D
                                                                 |___Ranking can be like C A D B

If winners of the semi-finals are A and D,
                                          |__If the winner is A,
                                          |                      |___Ranking can be like A D B C
                                          |                      |___Ranking can be like A D C B
                                          |
                                          |__If the winner is D,
                                                                 |___Ranking can be like D A B C
                                                                 |___Ranking can be like D A C B

If winners of the semi-finals are B and C,
                                          |__If the winner is B,
                                          |                      |___Ranking can be like B C A D
                                          |                      |___Ranking can be like B C D A
                                          |
                                          |__If the winner is C,
                                                                 |___Ranking can be like C B A D
                                                                 |___Ranking can be like C B D A

If winners of the semi-finals are B and D,
                                          |__If the winner is B,
                                          |                      |___Ranking can be like B D A C
                                          |                      |___Ranking can be like B D C A
                                          |
                                          |__If the winner is D,
                                                                 |___Ranking can be like D B A C
                                                                 |___Ranking can be like D B C A


For the second match-up, ---------------------------------------------------------------------
('A', 'C') ['B', 'D']

If winners of the semi-finals are A and B,
                                          |__If the winner is A,
                                          |                      |___Ranking can be like A B C D
                                          |                      |___Ranking can be like A B D C
                                          |
                                          |__If the winner is B,
                                                                 |___Ranking can be like B A C D
                                                                 |___Ranking can be like B A D C

If winners of the semi-finals are A and D,
                                          |__If the winner is A,
                                          |                      |___Ranking can be like A D C B
                                          |                      |___Ranking can be like A D B C
                                          |
                                          |__If the winner is D,
                                                                 |___Ranking can be like D A C B
                                                                 |___Ranking can be like D A B C

If winners of the semi-finals are C and B,
                                          |__If the winner is C,
                                          |                      |___Ranking can be like C B A D
                                          |                      |___Ranking can be like C B D A
                                          |
                                          |__If the winner is B,
                                                                 |___Ranking can be like B C A D
                                                                 |___Ranking can be like B C D A

If winners of the semi-finals are C and D,
                                          |__If the winner is C,
                                          |                      |___Ranking can be like C D A B
                                          |                      |___Ranking can be like C D B A
                                          |
                                          |__If the winner is D,
                                                                 |___Ranking can be like D C A B
                                                                 |___Ranking can be like D C B A
```

```
For the third match-up, ----------------------------------------------------------------------
('A', 'D') ['B', 'C']

If winners of the semi-finals are A and B,
                                          |__If the winner is A,
                                          |                     |___Ranking can be like A B C D
                                          |                     |___Ranking can be like A B D C
                                          |
                                          |__If the winner is B,
                                                                |___Ranking can be like B A C D
                                                                |___Ranking can be like B A D C

If winners of the semi-finals are A and C,
                                          |__If the winner is A,
                                          |                     |___Ranking can be like A C D B
                                          |                     |___Ranking can be like A C B D
                                          |
                                          |__If the winner is C,
                                                                |___Ranking can be like C A D B
                                                                |___Ranking can be like C A B D

If winners of the semi-finals are D and B,
                                          |__If the winner is D,
                                          |                     |___Ranking can be like D B A C
                                          |                     |___Ranking can be like D B C A
                                          |
                                          |__If the winner is B,
                                                                |___Ranking can be like B D A C
                                                                |___Ranking can be like B D C A

If winners of the semi-finals are D and C,
                                          |__If the winner is D,
                                          |                     |___Ranking can be like D C A B
                                          |                     |___Ranking can be like D C B A
                                          |
                                          |__If the winner is C,
                                                                |___Ranking can be like C D A B
                                                                |___Ranking can be like C D B A
```

Appendix 2

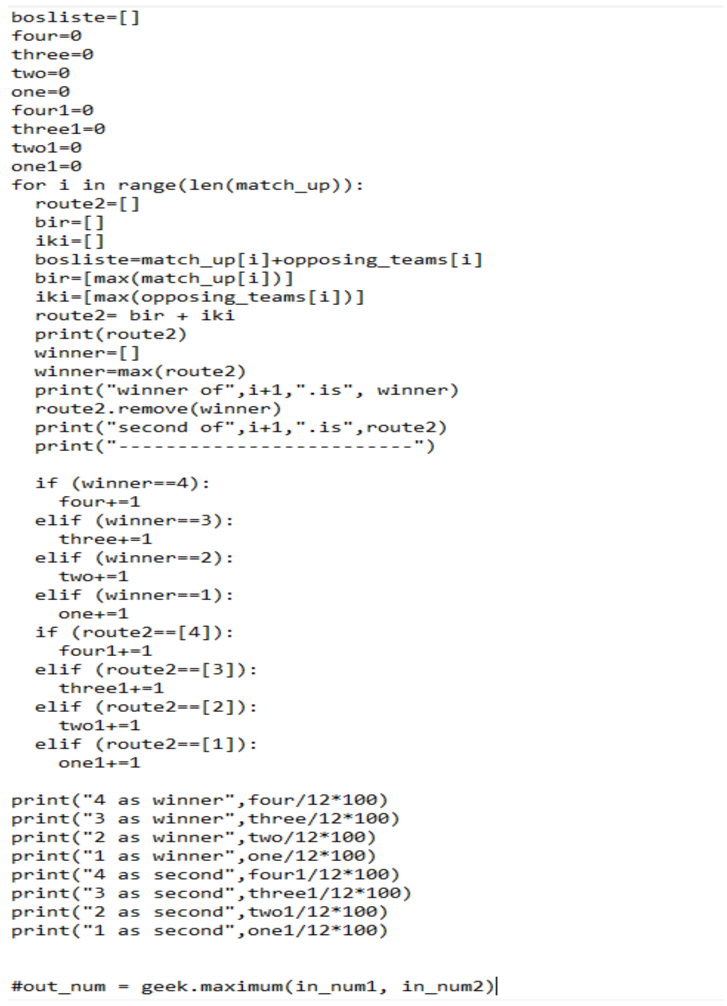
```
import itertools
participants =["A","B","C","D"]
route1=[list(itertools.combinations(participants, 2))]
for i in participants:
  print("1.elenen: ", i )
  participants2= participants.copy()
  participants2.remove(i)
  remainings= list(participants2)
  route2= [list(itertools.combinations(remainings,2))]

  for k in remainings:
    #print("2.elenen: ", k)
    remainings2= remainings.copy()
    remainings2.remove(k)
    remains_2nd= list(remainings2)
    route3= [list(itertools.combinations(remains_2nd,2))]


    for l in remains_2nd:
      #print("kazanan: ", l)
      winner= [l]
      possible= route1 + route2 + route3 + winner
      print(possible)
      winner= []



    remains_2nd= []

  remainings=[]
```

Appendix 2 Output



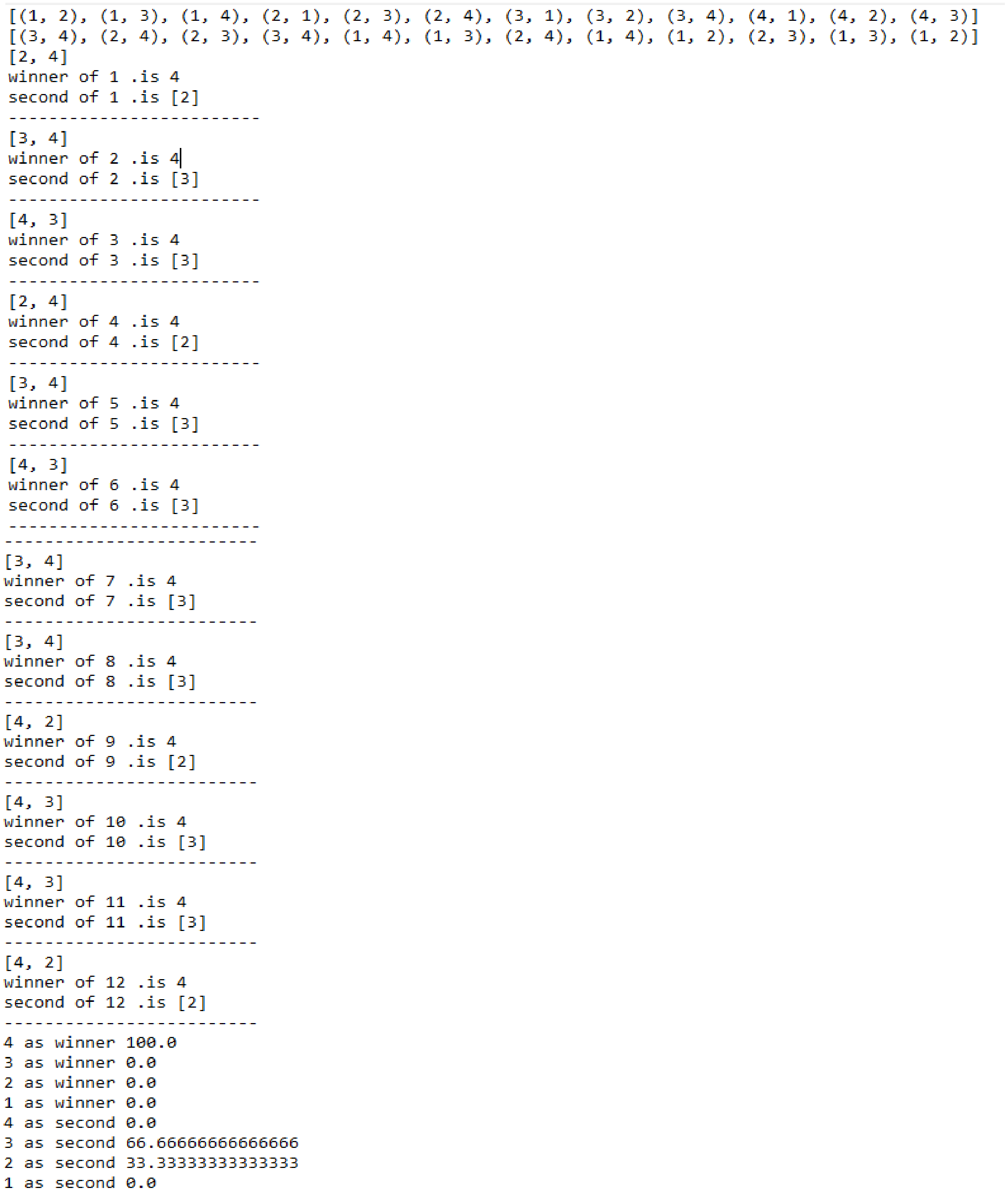
```
1.elenen:  A
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('C', 'D')], 'C']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('C', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'D')], 'B']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C')], 'B']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C'), ('B', 'D'), ('C', 'D')], [('B', 'C')], 'C']
1.elenen:  B
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('C', 'D')], 'C']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('C', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('A', 'D')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('A', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('A', 'C')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'C'), ('A', 'D'), ('C', 'D')], [('A', 'C')], 'C']
1.elenen:  C
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('B', 'D')], 'B']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('B', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('A', 'D')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('A', 'D')], 'D']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('A', 'B')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'D'), ('B', 'D')], [('A', 'B')], 'B']
1.elenen:  D
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('B', 'C')], 'B']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('B', 'C')], 'C']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('A', 'C')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('A', 'C')], 'C']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('A', 'B')], 'A']
[[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')], [('A', 'B'), ('A', 'C'), ('B', 'C')], [('A', 'B')], 'B']
```

Appendix 3



```
bosliste=[]
four=0
three=0
two=0
one=0
four1=0
three1=0
two1=0
one1=0
for i in range(len(match_up)):
  route2=[]
  bir=[]
  iki=[]
  bosliste=match_up[i]+opposing_teams[i]
  bir=[max(match_up[i])]
  iki=[max(opposing_teams[i])]
  route2= bir + iki
  print(route2)
  winner=[]
  winner=max(route2)
  print("winner of",i+1,".is", winner)
  route2.remove(winner)
  print("second of",i+1,".is",route2)
  print("-------------------------")

  if (winner==4):
    four+=1
  elif (winner==3):
    three+=1
  elif (winner==2):
    two+=1
  elif (winner==1):
    one+=1
  if (route2==[4]):
    four1+=1
  elif (route2==[3]):
    three1+=1
  elif (route2==[2]):
    two1+=1
  elif (route2==[1]):
    one1+=1

print("4 as winner",four/12*100)
print("3 as winner",three/12*100)
print("2 as winner",two/12*100)
print("1 as winner",one/12*100)
print("4 as second",four1/12*100)
print("3 as second",three1/12*100)
print("2 as second",two1/12*100)
print("1 as second",one1/12*100)


#out_num = geek.maximum(in_num1, in_num2)
```

Appendix 3 Output

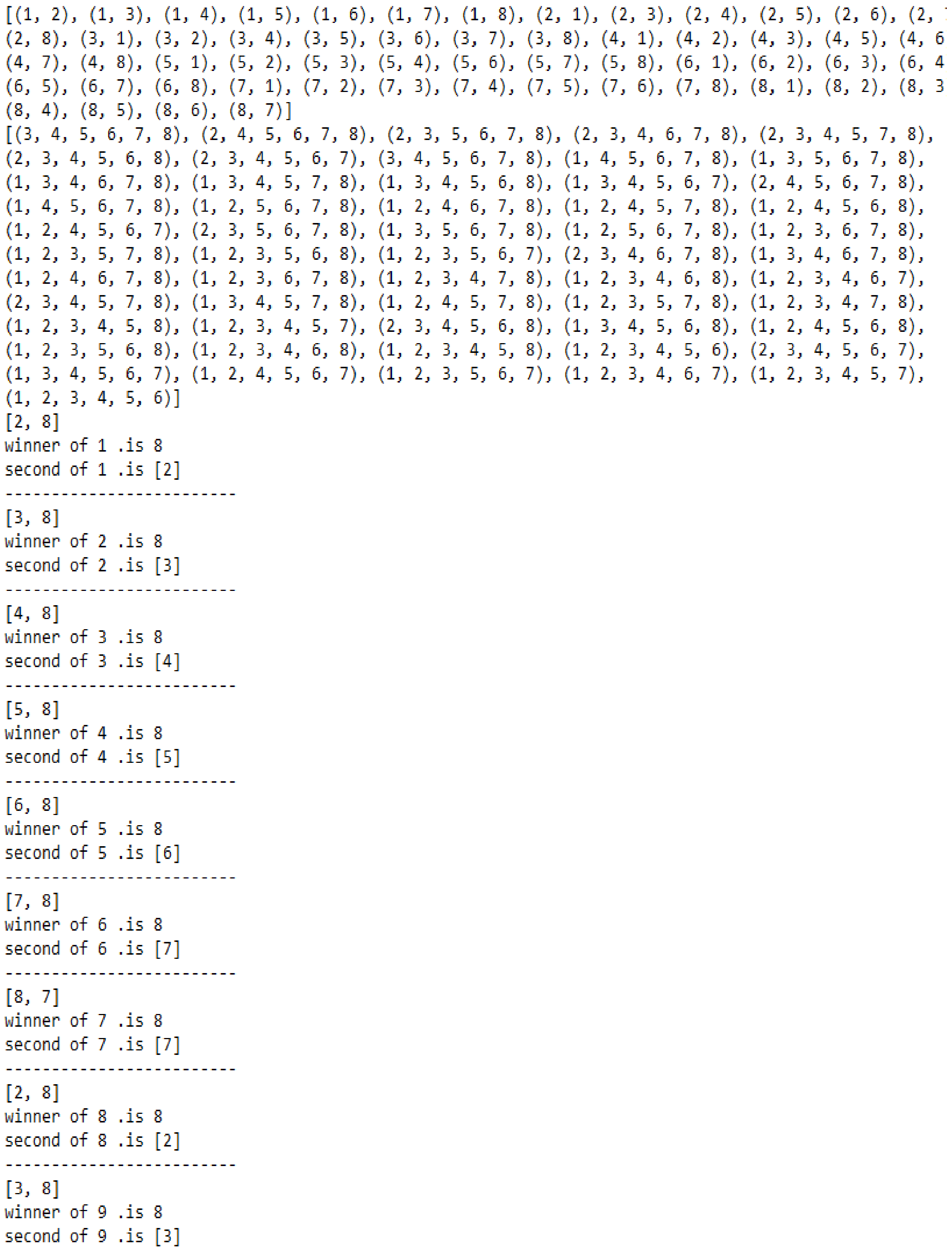


```
[(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3)]
[(3, 4), (2, 4), (2, 3), (3, 4), (1, 4), (1, 3), (2, 4), (1, 4), (1, 2), (2, 3), (1, 3), (1, 2)]
[2, 4]
winner of 1 .is 4
second of 1 .is [2]
-------------------------
[3, 4]
winner of 2 .is 4
second of 2 .is [3]
-------------------------
[4, 3]
winner of 3 .is 4
second of 3 .is [3]
-------------------------
[2, 4]
winner of 4 .is 4
second of 4 .is [2]
-------------------------
[3, 4]
winner of 5 .is 4
second of 5 .is [3]
-------------------------
[4, 3]
winner of 6 .is 4
second of 6 .is [3]
-------------------------
-------------------------
[3, 4]
winner of 7 .is 4
second of 7 .is [3]
-------------------------
[3, 4]
winner of 8 .is 4
second of 8 .is [3]
-------------------------
[4, 2]
winner of 9 .is 4
second of 9 .is [2]
-------------------------
[4, 3]
winner of 10 .is 4
second of 10 .is [3]
-------------------------
[4, 3]
winner of 11 .is 4
second of 11 .is [3]
-------------------------
[4, 2]
winner of 12 .is 4
second of 12 .is [2]
-------------------------
4 as winner 100.0
3 as winner 0.0
2 as winner 0.0
1 as winner 0.0
4 as second 0.0
3 as second 66.66666666666666
2 as second 33.33333333333333
1 as second 0.0
```

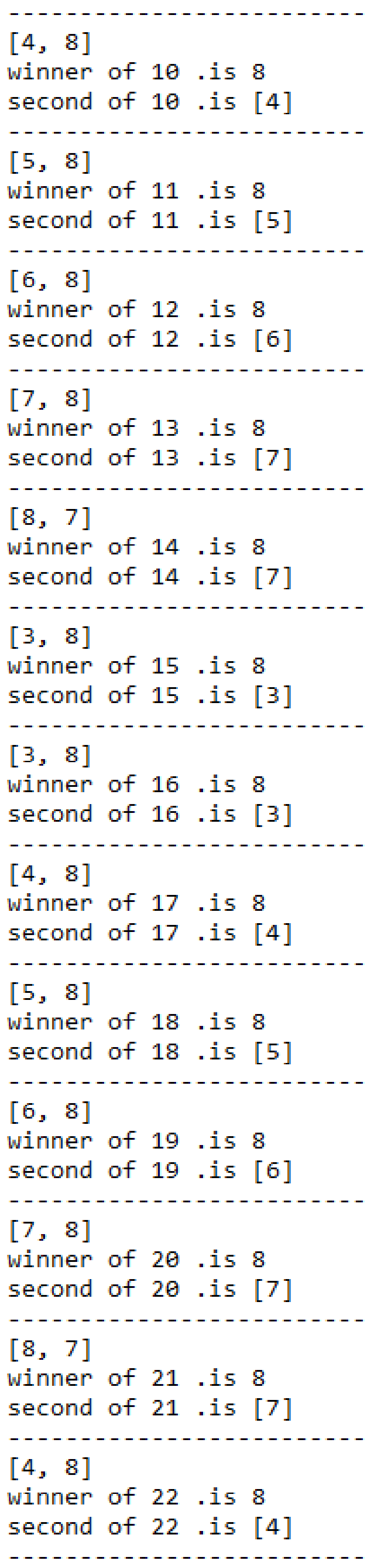
Appendix 4

```
import itertools
#list(itertools.permutations([1,2,3,4], 2))
teams = [1,2,3,4,5,6,7,8]
match_up = list(itertools.permutations(teams, 2))

opposing_teams=[]
for current_team in match_up:
  opposing_team=[]
  for team in teams:
    if team not in current_team:
      opposing_team.append(team)
  opposing_teams.append(tuple(opposing_team))

print(match_up)
print(opposing_teams)

#dict(zip(match_up, opposing_teams))
bosliste=[]
x=0
for i in range(len(match_up)):
  route2=[]
  bir=[]
  iki=[]
  bosliste=match_up[i]+opposing_teams[i]
  bir=[max(match_up[i])]
  iki=[max(opposing_teams[i])]
  route2= bir + iki
  print(route2)
  winner=[]
  winner=max(route2)
  print("winner of",i+1,".is", winner)
  route2.remove(winner)
  print("second of",i+1,".is",route2)
  print("------------------------")
  if route2== [7]:
    x+=1
print("7 as second", x/len(match_up)*100)
print("7 as second", x/56*100)
```

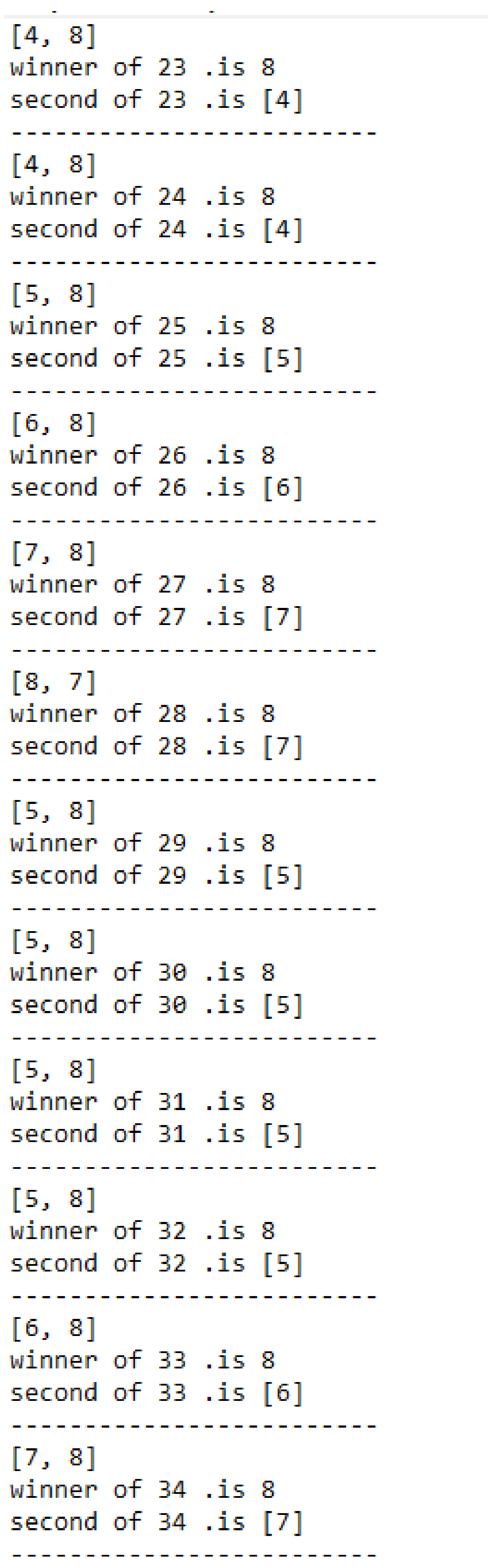
Appendix 4 Output



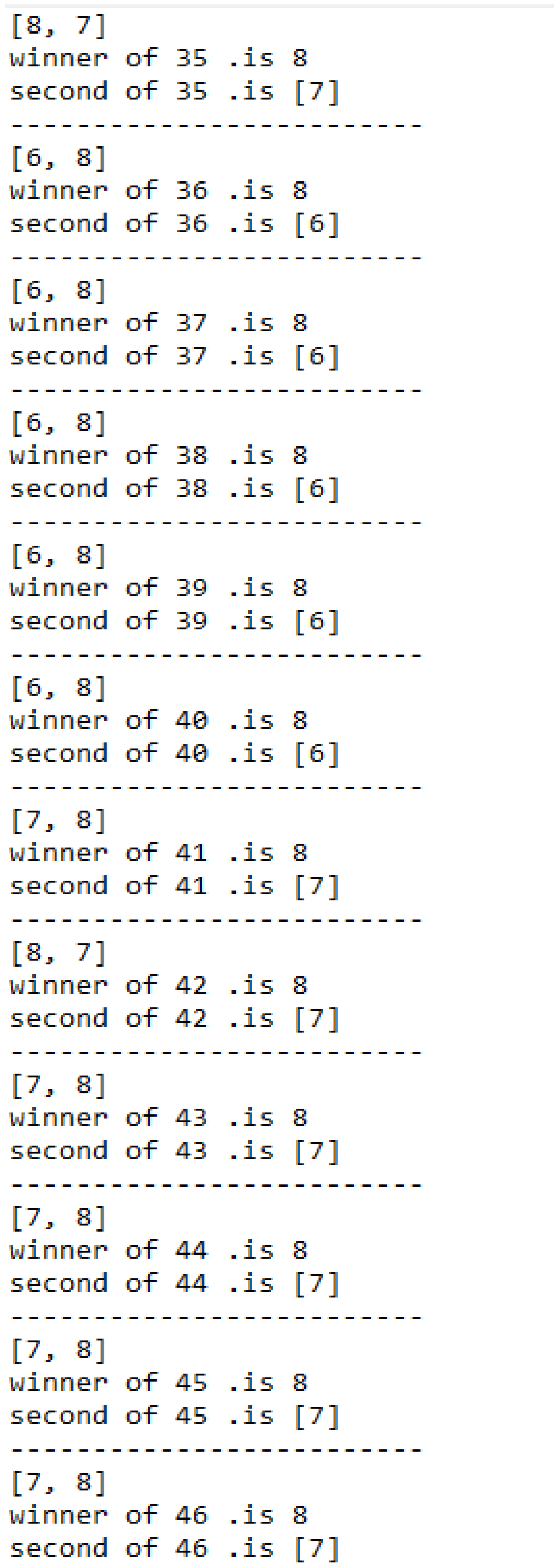
```
[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (2, 1), (2, 3), (2, 4), (2, 5), (2, 6), (2,
(2, 8), (3, 1), (3, 2), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (4, 1), (4, 2), (4, 3), (4, 5), (4, 6
(4, 7), (4, 8), (5, 1), (5, 2), (5, 3), (5, 4), (5, 6), (5, 7), (5, 8), (6, 1), (6, 2), (6, 3), (6, 4
(6, 5), (6, 7), (6, 8), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 8), (8, 1), (8, 2), (8, 3
(8, 4), (8, 5), (8, 6), (8, 7)]
[(3, 4, 5, 6, 7, 8), (2, 4, 5, 6, 7, 8), (2, 3, 5, 6, 7, 8), (2, 3, 4, 6, 7, 8), (2, 3, 4, 5, 7, 8),
(2, 3, 4, 5, 6, 8), (2, 3, 4, 5, 6, 7), (3, 4, 5, 6, 7, 8), (1, 4, 5, 6, 7, 8), (1, 3, 5, 6, 7, 8),
(1, 3, 4, 6, 7, 8), (1, 3, 4, 5, 7, 8), (1, 3, 4, 5, 6, 8), (1, 3, 4, 5, 6, 7), (2, 4, 5, 6, 7, 8),
(1, 4, 5, 6, 7, 8), (1, 2, 5, 6, 7, 8), (1, 2, 4, 6, 7, 8), (1, 2, 4, 5, 7, 8), (1, 2, 4, 5, 6, 8),
(1, 2, 4, 5, 6, 7), (2, 3, 5, 6, 7, 8), (1, 3, 5, 6, 7, 8), (1, 2, 5, 6, 7, 8), (1, 2, 3, 6, 7, 8),
(1, 2, 3, 5, 7, 8), (1, 2, 3, 5, 6, 8), (1, 2, 3, 5, 6, 7), (2, 3, 4, 6, 7, 8), (1, 3, 4, 6, 7, 8),
(1, 2, 4, 6, 7, 8), (1, 2, 3, 6, 7, 8), (1, 2, 3, 4, 7, 8), (1, 2, 3, 4, 6, 8), (1, 2, 3, 4, 6, 7),
(2, 3, 4, 5, 7, 8), (1, 3, 4, 5, 7, 8), (1, 2, 4, 5, 7, 8), (1, 2, 3, 5, 7, 8), (1, 2, 3, 4, 7, 8),
(1, 2, 3, 4, 5, 8), (1, 2, 3, 4, 5, 7), (2, 3, 4, 5, 6, 8), (1, 3, 4, 5, 6, 8), (1, 2, 4, 5, 6, 8),
(1, 2, 3, 5, 6, 8), (1, 2, 3, 4, 6, 8), (1, 2, 3, 4, 5, 8), (1, 2, 3, 4, 5, 6), (2, 3, 4, 5, 6, 7),
(1, 3, 4, 5, 6, 7), (1, 2, 4, 5, 6, 7), (1, 2, 3, 5, 6, 7), (1, 2, 3, 4, 6, 7), (1, 2, 3, 4, 5, 7),
(1, 2, 3, 4, 5, 6)]
[2, 8]
winner of 1 .is 8
second of 1 .is [2]
-------------------------
[3, 8]
winner of 2 .is 8
second of 2 .is [3]
-------------------------
[4, 8]
winner of 3 .is 8
second of 3 .is [4]
-------------------------
[5, 8]
winner of 4 .is 8
second of 4 .is [5]
-------------------------
[6, 8]
winner of 5 .is 8
second of 5 .is [6]
-------------------------
[7, 8]
winner of 6 .is 8
second of 6 .is [7]
-------------------------
[8, 7]
winner of 7 .is 8
second of 7 .is [7]
-------------------------
[2, 8]
winner of 8 .is 8
second of 8 .is [2]
-------------------------
[3, 8]
winner of 9 .is 8
second of 9 .is [3]
```



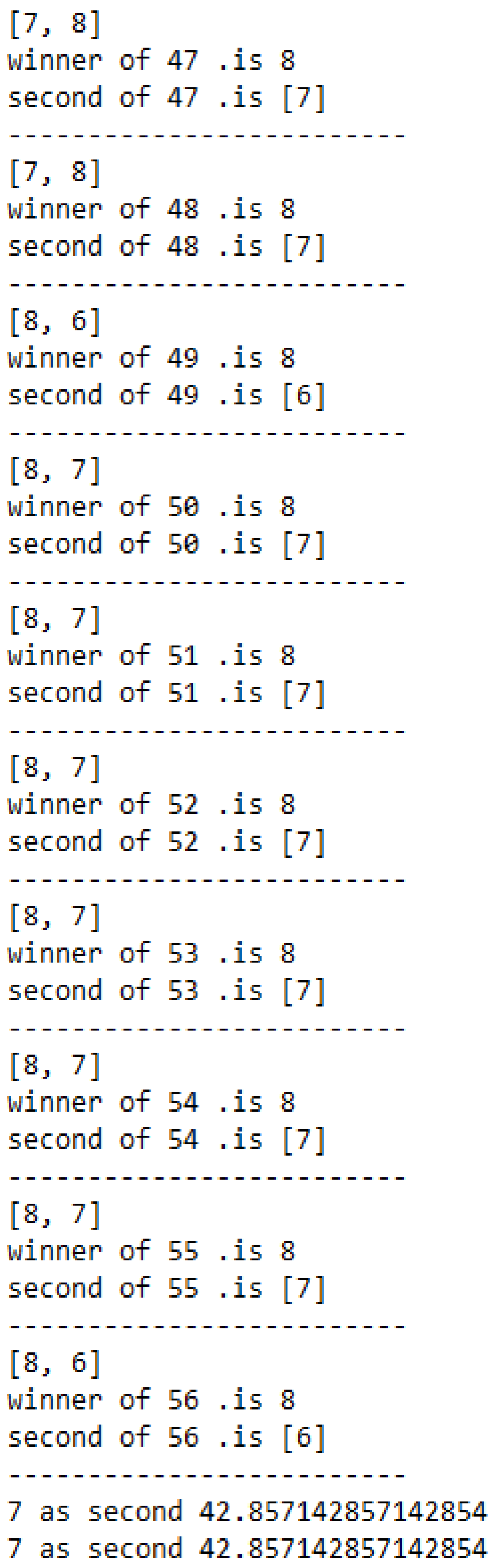
```
-------------------------
[4, 8]
winner of 10 .is 8
second of 10 .is [4]
-------------------------
[5, 8]
winner of 11 .is 8
second of 11 .is [5]
-------------------------
[6, 8]
winner of 12 .is 8
second of 12 .is [6]
-------------------------
[7, 8]
winner of 13 .is 8
second of 13 .is [7]
-------------------------
[8, 7]
winner of 14 .is 8
second of 14 .is [7]
-------------------------
[3, 8]
winner of 15 .is 8
second of 15 .is [3]
-------------------------
[3, 8]
winner of 16 .is 8
second of 16 .is [3]
-------------------------
[4, 8]
winner of 17 .is 8
second of 17 .is [4]
-------------------------
[5, 8]
winner of 18 .is 8
second of 18 .is [5]
-------------------------
[6, 8]
winner of 19 .is 8
second of 19 .is [6]
-------------------------
[7, 8]
winner of 20 .is 8
second of 20 .is [7]
-------------------------
[8, 7]
winner of 21 .is 8
second of 21 .is [7]
-------------------------
[4, 8]
winner of 22 .is 8
second of 22 .is [4]
-------------------------
```



```
[4, 8]
winner of 23 .is 8
second of 23 .is [4]
-------------------------
[4, 8]
winner of 24 .is 8
second of 24 .is [4]
-------------------------
[5, 8]
winner of 25 .is 8
second of 25 .is [5]
-------------------------
[6, 8]
winner of 26 .is 8
second of 26 .is [6]
-------------------------
[7, 8]
winner of 27 .is 8
second of 27 .is [7]
-------------------------
[8, 7]
winner of 28 .is 8
second of 28 .is [7]
-------------------------
[5, 8]
winner of 29 .is 8
second of 29 .is [5]
-------------------------
[5, 8]
winner of 30 .is 8
second of 30 .is [5]
-------------------------
[5, 8]
winner of 31 .is 8
second of 31 .is [5]
-------------------------
[5, 8]
winner of 32 .is 8
second of 32 .is [5]
-------------------------
[6, 8]
winner of 33 .is 8
second of 33 .is [6]
-------------------------
[7, 8]
winner of 34 .is 8
second of 34 .is [7]
-------------------------
```

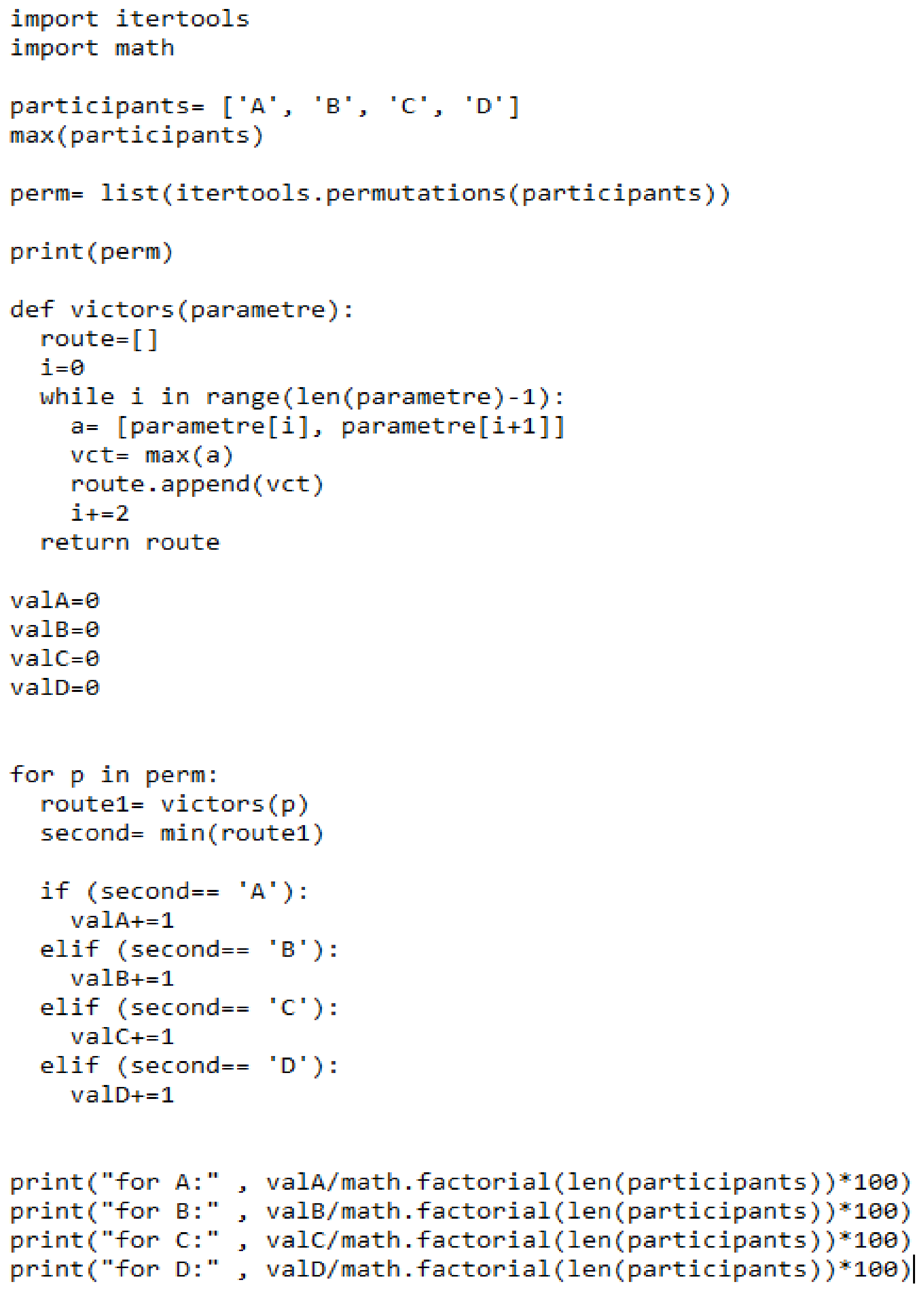


```
[8, 7]
winner of 35 .is 8
second of 35 .is [7]
-------------------------
[6, 8]
winner of 36 .is 8
second of 36 .is [6]
-------------------------
[6, 8]
winner of 37 .is 8
second of 37 .is [6]
-------------------------
[6, 8]
winner of 38 .is 8
second of 38 .is [6]
-------------------------
[6, 8]
winner of 39 .is 8
second of 39 .is [6]
-------------------------
[6, 8]
winner of 40 .is 8
second of 40 .is [6]
-------------------------
[7, 8]
winner of 41 .is 8
second of 41 .is [7]
-------------------------
[8, 7]
winner of 42 .is 8
second of 42 .is [7]
-------------------------
[7, 8]
winner of 43 .is 8
second of 43 .is [7]
-------------------------
[7, 8]
winner of 44 .is 8
second of 44 .is [7]
-------------------------
[7, 8]
winner of 45 .is 8
second of 45 .is [7]
-------------------------
[7, 8]
winner of 46 .is 8
second of 46 .is [7]
```



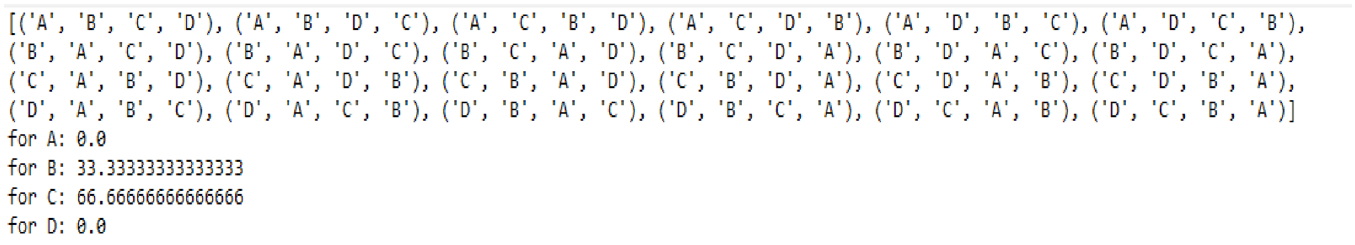
```
[7, 8]
winner of 47 .is 8
second of 47 .is [7]
-------------------------
[7, 8]
winner of 48 .is 8
second of 48 .is [7]
-------------------------
[8, 6]
winner of 49 .is 8
second of 49 .is [6]
-------------------------
[8, 7]
winner of 50 .is 8
second of 50 .is [7]
-------------------------
[8, 7]
winner of 51 .is 8
second of 51 .is [7]
-------------------------
[8, 7]
winner of 52 .is 8
second of 52 .is [7]
-------------------------
[8, 7]
winner of 53 .is 8
second of 53 .is [7]
-------------------------
[8, 7]
winner of 54 .is 8
second of 54 .is [7]
-------------------------
[8, 7]
winner of 55 .is 8
second of 55 .is [7]
-------------------------
[8, 6]
winner of 56 .is 8
second of 56 .is [6]
-------------------------
7 as second 42.857142857142854
7 as second 42.857142857142854
```

Appendix 5



```
import itertools
import math

participants= ['A', 'B', 'C', 'D']
max(participants)

perm= list(itertools.permutations(participants))

print(perm)

def victors(parametre):
  route=[]
  i=0
  while i in range(len(parametre)-1):
    a= [parametre[i], parametre[i+1]]
    vct= max(a)
    route.append(vct)
    i+=2
  return route

valA=0
valB=0
valC=0
valD=0


for p in perm:
  route1= victors(p)
  second= min(route1)

  if (second== 'A'):
    valA+=1
  elif (second== 'B'):
    valB+=1
  elif (second== 'C'):
    valC+=1
  elif (second== 'D'):
    valD+=1


print("for A:" , valA/math.factorial(len(participants))*100)
print("for B:" , valB/math.factorial(len(participants))*100)
print("for C:" , valC/math.factorial(len(participants))*100)
print("for D:" , valD/math.factorial(len(participants))*100)
```

Appendix 5 Output

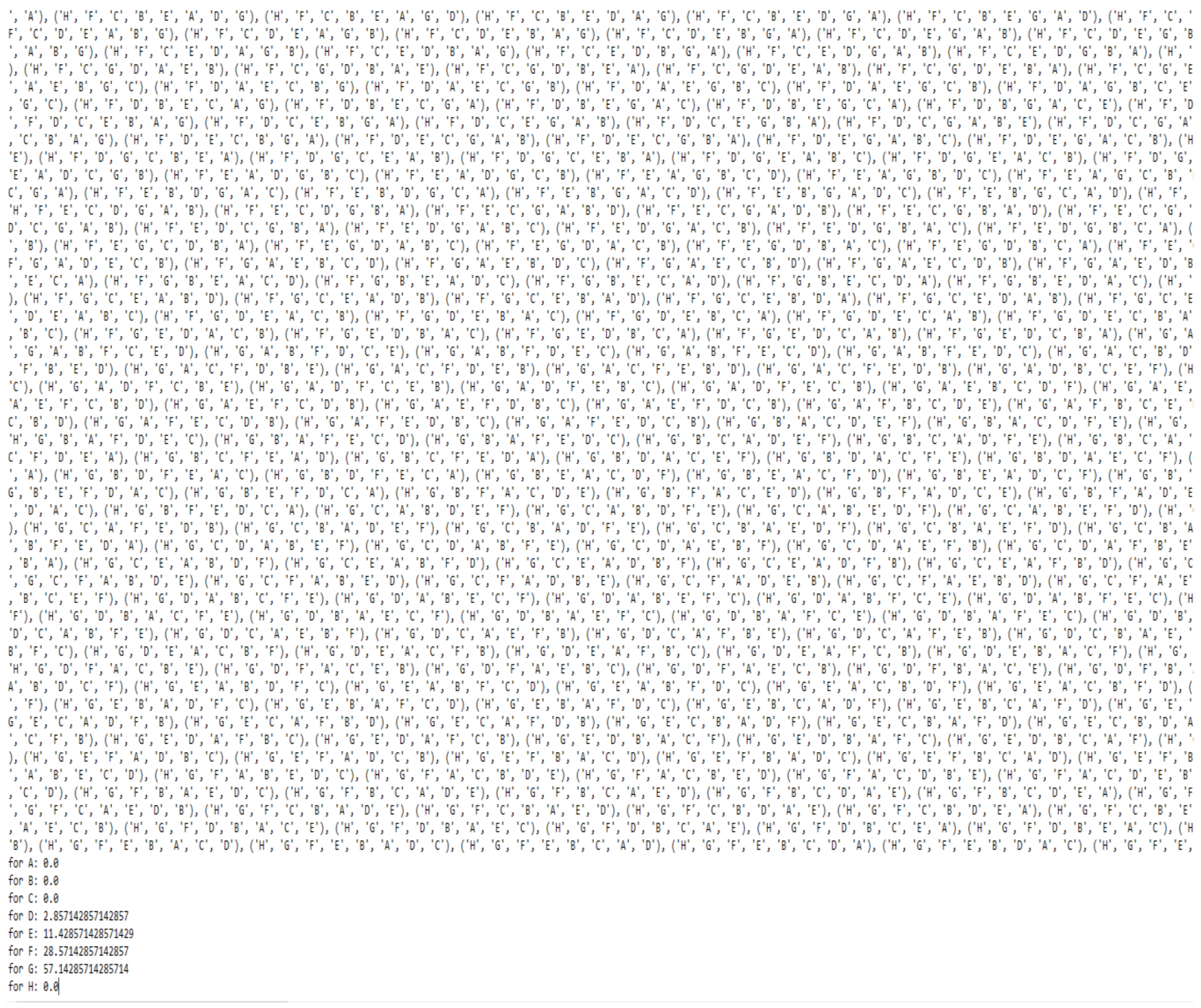


```
[('A', 'B', 'C', 'D'), ('A', 'B', 'D', 'C'), ('A', 'C', 'B', 'D'), ('A', 'C', 'D', 'B'), ('A', 'D', 'B', 'C'), ('A', 'D', 'C', 'B'),
('B', 'A', 'C', 'D'), ('B', 'A', 'D', 'C'), ('B', 'C', 'A', 'D'), ('B', 'C', 'D', 'A'), ('B', 'D', 'A', 'C'), ('B', 'D', 'C', 'A'),
('C', 'A', 'B', 'D'), ('C', 'A', 'D', 'B'), ('C', 'B', 'A', 'D'), ('C', 'B', 'D', 'A'), ('C', 'D', 'A', 'B'), ('C', 'D', 'B', 'A'),
('D', 'A', 'B', 'C'), ('D', 'A', 'C', 'B'), ('D', 'B', 'A', 'C'), ('D', 'B', 'C', 'A'), ('D', 'C', 'A', 'B'), ('D', 'C', 'B', 'A')]
for A: 0.0
for B: 33.33333333333333
for C: 66.66666666666666
for D: 0.0
```

Appendix 6

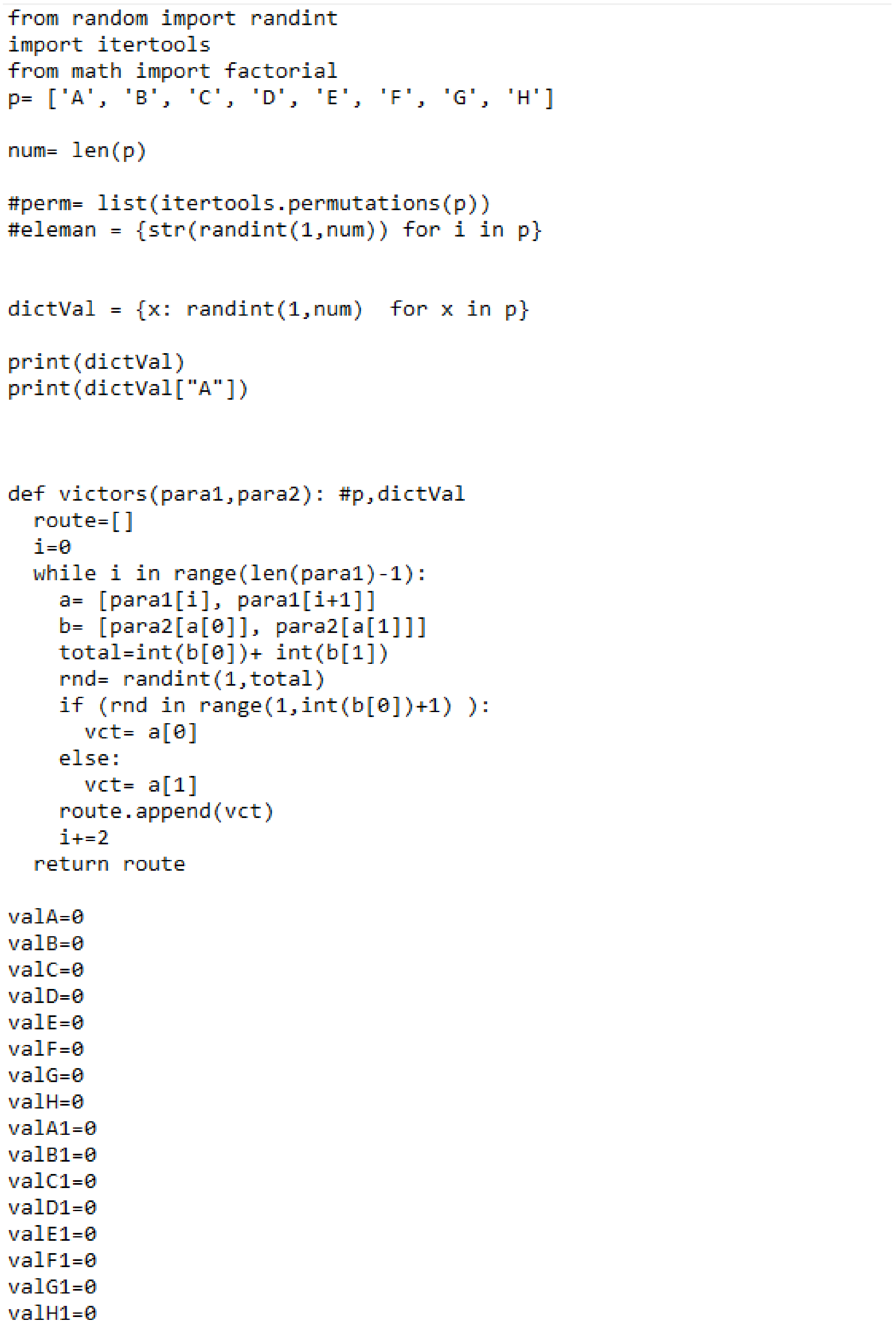
```
participants= ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']
max(participants)

perm= list(itertools.permutations(participants))

print(perm)

def victors(parametre):
  route=[]
  i=0
  while i in range(len(parametre)-1):
    a= [parametre[i], parametre[i+1]]
    vct= max(a)
    route.append(vct)
    i+=2
  return route

valA=0
valB=0
valC=0
valD=0
valE=0
valF=0
valG=0
valH=0

for p in perm:
  route1= victors(p)
  route2= victors(route1)
  second= min(route2)

  if (second== 'A'):
    valA+=1
  elif (second== 'B'):
    valB+=1
  elif (second== 'C'):
    valC+=1
  elif (second== 'D'):
    valD+=1
  elif (second== 'E'):
    valE+=1
  elif (second== 'F'):
    valF+=1
  elif (second== 'G'):
    valG+=1
  elif (second== 'H'):
    valH+=1

print("for A:" , valA/math.factorial(len(participants))*100)
print("for B:" , valB/math.factorial(len(participants))*100)
print("for C:" , valC/math.factorial(len(participants))*100)
print("for D:" , valD/math.factorial(len(participants))*100)
print("for E:" , valE/math.factorial(len(participants))*100)
print("for F:" , valF/math.factorial(len(participants))*100)
print("for G:" , valG/math.factorial(len(participants))*100)
print("for H:" , valH/math.factorial(len(participants))*100)
```

Appendix 6 Output

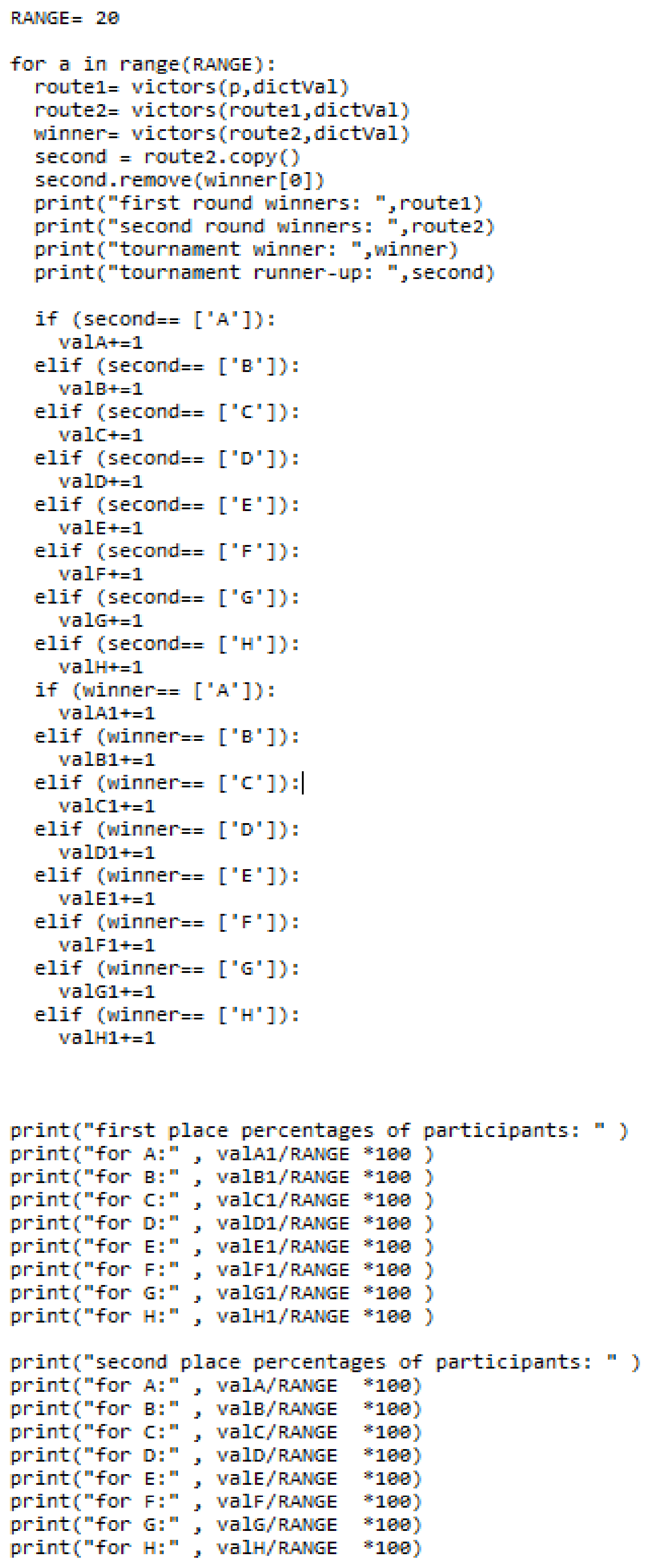


```
', 'A'), ('H', 'F', 'C', 'B', 'E', 'A', 'D', 'G'), ('H', 'F', 'C', 'B', 'E', 'A', 'G', 'D'), ('H', 'F', 'C', 'B', 'E', 'D', 'A', 'G'), ('H', 'F', 'C', 'B', 'E', 'D', 'G', 'A'), ('H', 'F', 'C', 'B', 'E', 'G', 'A', 'D'), ('H', 'F', 'C', '
F', 'C', 'D', 'E', 'A', 'B', 'G'), ('H', 'F', 'C', 'D', 'E', 'A', 'G', 'B'), ('H', 'F', 'C', 'D', 'E', 'B', 'A', 'G'), ('H', 'F', 'C', 'D', 'E', 'B', 'G', 'A'), ('H', 'F', 'C', 'D', 'E', 'G', 'A', 'B'), ('H', 'F', 'C', 'D', 'E', 'G', 'B
', 'A', 'B', 'G'), ('H', 'F', 'C', 'E', 'D', 'A', 'G', 'B'), ('H', 'F', 'C', 'E', 'D', 'B', 'A', 'G'), ('H', 'F', 'C', 'E', 'D', 'B', 'G', 'A'), ('H', 'F', 'C', 'E', 'D', 'G', 'A', 'B'), ('H', 'F', 'C', 'E', 'D', 'G', 'B', 'A'), ('H', '
), ('H', 'F', 'C', 'G', 'D', 'A', 'E', 'B'), ('H', 'F', 'C', 'G', 'D', 'B', 'A', 'E'), ('H', 'F', 'C', 'G', 'D', 'B', 'E', 'A'), ('H', 'F', 'C', 'G', 'D', 'E', 'A', 'B'), ('H', 'F', 'C', 'G', 'D', 'E', 'B', 'A'), ('H', 'F', 'C', 'G', 'E
', 'A', 'E', 'B', 'G', 'C'), ('H', 'F', 'D', 'A', 'E', 'C', 'B', 'G'), ('H', 'F', 'D', 'A', 'E', 'C', 'G', 'B'), ('H', 'F', 'D', 'A', 'E', 'G', 'B', 'C'), ('H', 'F', 'D', 'A', 'E', 'G', 'C', 'B'), ('H', 'F', 'D', 'A', 'G', 'B', 'C', 'E'
, 'G', 'C'), ('H', 'F', 'D', 'B', 'E', 'C', 'A', 'G'), ('H', 'F', 'D', 'B', 'E', 'C', 'G', 'A'), ('H', 'F', 'D', 'B', 'E', 'G', 'A', 'C'), ('H', 'F', 'D', 'B', 'E', 'G', 'C', 'A'), ('H', 'F', 'D', 'B', 'G', 'A', 'C', 'E'), ('H', 'F', 'D
', 'F', 'D', 'C', 'E', 'B', 'A', 'G'), ('H', 'F', 'D', 'C', 'E', 'B', 'G', 'A'), ('H', 'F', 'D', 'C', 'E', 'G', 'A', 'B'), ('H', 'F', 'D', 'C', 'E', 'G', 'B', 'A'), ('H', 'F', 'D', 'C', 'G', 'A', 'B', 'E'), ('H', 'F', 'D', 'C', 'G', 'A'
, 'C', 'B', 'A', 'G'), ('H', 'F', 'D', 'E', 'C', 'B', 'G', 'A'), ('H', 'F', 'D', 'E', 'C', 'G', 'A', 'B'), ('H', 'F', 'D', 'E', 'C', 'G', 'B', 'A'), ('H', 'F', 'D', 'E', 'G', 'A', 'B', 'C'), ('H', 'F', 'D', 'E', 'G', 'A', 'C', 'B'), ('H
'E'), ('H', 'F', 'D', 'G', 'C', 'B', 'E', 'A'), ('H', 'F', 'D', 'G', 'C', 'E', 'A', 'B'), ('H', 'F', 'D', 'G', 'C', 'E', 'B', 'A'), ('H', 'F', 'D', 'G', 'E', 'A', 'B', 'C'), ('H', 'F', 'D', 'G', 'E', 'A', 'C', 'B'), ('H', 'F', 'D', 'G',
'E', 'A', 'D', 'C', 'G', 'B'), ('H', 'F', 'E', 'A', 'D', 'G', 'B', 'C'), ('H', 'F', 'E', 'A', 'D', 'G', 'C', 'B'), ('H', 'F', 'E', 'A', 'G', 'B', 'C', 'D'), ('H', 'F', 'E', 'A', 'G', 'B', 'D', 'C'), ('H', 'F', 'E', 'A', 'G', 'C', 'B', '
C', 'G', 'A'), ('H', 'F', 'E', 'B', 'D', 'G', 'A', 'C'), ('H', 'F', 'E', 'B', 'D', 'G', 'C', 'A'), ('H', 'F', 'E', 'B', 'G', 'A', 'C', 'D'), ('H', 'F', 'E', 'B', 'G', 'A', 'D', 'C'), ('H', 'F', 'E', 'B', 'G', 'C', 'A', 'D'), ('H', 'F',
'H', 'F', 'E', 'C', 'D', 'G', 'A', 'B'), ('H', 'F', 'E', 'C', 'D', 'G', 'B', 'A'), ('H', 'F', 'E', 'C', 'G', 'A', 'B', 'D'), ('H', 'F', 'E', 'C', 'G', 'A', 'D', 'B'), ('H', 'F', 'E', 'C', 'G', 'B', 'A', 'D'), ('H', 'F', 'E', 'C', 'G', '
D', 'C', 'G', 'A', 'B'), ('H', 'F', 'E', 'D', 'C', 'G', 'B', 'A'), ('H', 'F', 'E', 'D', 'G', 'A', 'B', 'C'), ('H', 'F', 'E', 'D', 'G', 'A', 'C', 'B'), ('H', 'F', 'E', 'D', 'G', 'B', 'A', 'C'), ('H', 'F', 'E', 'D', 'G', 'B', 'C', 'A'), (
', 'B'), ('H', 'F', 'E', 'G', 'C', 'D', 'B', 'A'), ('H', 'F', 'E', 'G', 'D', 'A', 'B', 'C'), ('H', 'F', 'E', 'G', 'D', 'A', 'C', 'B'), ('H', 'F', 'E', 'G', 'D', 'B', 'A', 'C'), ('H', 'F', 'E', 'G', 'D', 'B', 'C', 'A'), ('H', 'F', 'E', '
F', 'G', 'A', 'D', 'E', 'C', 'B'), ('H', 'F', 'G', 'A', 'E', 'B', 'C', 'D'), ('H', 'F', 'G', 'A', 'E', 'B', 'D', 'C'), ('H', 'F', 'G', 'A', 'E', 'C', 'B', 'D'), ('H', 'F', 'G', 'A', 'E', 'C', 'D', 'B'), ('H', 'F', 'G', 'A', 'E', 'D', 'B
', 'E', 'C', 'A'), ('H', 'F', 'G', 'B', 'E', 'A', 'C', 'D'), ('H', 'F', 'G', 'B', 'E', 'A', 'D', 'C'), ('H', 'F', 'G', 'B', 'E', 'C', 'A', 'D'), ('H', 'F', 'G', 'B', 'E', 'C', 'D', 'A'), ('H', 'F', 'G', 'B', 'E', 'D', 'A', 'C'), ('H', '
), ('H', 'F', 'G', 'C', 'E', 'A', 'B', 'D'), ('H', 'F', 'G', 'C', 'E', 'A', 'D', 'B'), ('H', 'F', 'G', 'C', 'E', 'B', 'A', 'D'), ('H', 'F', 'G', 'C', 'E', 'B', 'D', 'A'), ('H', 'F', 'G', 'C', 'E', 'D', 'A', 'B'), ('H', 'F', 'G', 'C', 'E
', 'D', 'E', 'A', 'B', 'C'), ('H', 'F', 'G', 'D', 'E', 'A', 'C', 'B'), ('H', 'F', 'G', 'D', 'E', 'B', 'A', 'C'), ('H', 'F', 'G', 'D', 'E', 'B', 'C', 'A'), ('H', 'F', 'G', 'D', 'E', 'C', 'A', 'B'), ('H', 'F', 'G', 'D', 'E', 'C', 'B', 'A'
, 'B', 'C'), ('H', 'F', 'G', 'E', 'D', 'A', 'C', 'B'), ('H', 'F', 'G', 'E', 'D', 'B', 'A', 'C'), ('H', 'F', 'G', 'E', 'D', 'B', 'C', 'A'), ('H', 'F', 'G', 'E', 'D', 'C', 'A', 'B'), ('H', 'F', 'G', 'E', 'D', 'C', 'B', 'A'), ('H', 'G', 'A
', 'G', 'A', 'B', 'F', 'C', 'E', 'D'), ('H', 'G', 'A', 'B', 'F', 'D', 'C', 'E'), ('H', 'G', 'A', 'B', 'F', 'D', 'E', 'C'), ('H', 'G', 'A', 'B', 'F', 'E', 'C', 'D'), ('H', 'G', 'A', 'B', 'F', 'E', 'D', 'C'), ('H', 'G', 'A', 'C', 'B', 'D'
, 'F', 'B', 'E', 'D'), ('H', 'G', 'A', 'C', 'F', 'D', 'B', 'E'), ('H', 'G', 'A', 'C', 'F', 'D', 'E', 'B'), ('H', 'G', 'A', 'C', 'F', 'E', 'B', 'D'), ('H', 'G', 'A', 'C', 'F', 'E', 'D', 'B'), ('H', 'G', 'A', 'D', 'B', 'C', 'E', 'F'), ('H
'C'), ('H', 'G', 'A', 'D', 'F', 'C', 'B', 'E'), ('H', 'G', 'A', 'D', 'F', 'C', 'E', 'B'), ('H', 'G', 'A', 'D', 'F', 'E', 'B', 'C'), ('H', 'G', 'A', 'D', 'F', 'E', 'C', 'B'), ('H', 'G', 'A', 'E', 'B', 'C', 'D', 'F'), ('H', 'G', 'A', 'E',
'A', 'E', 'F', 'C', 'B', 'D'), ('H', 'G', 'A', 'E', 'F', 'C', 'D', 'B'), ('H', 'G', 'A', 'E', 'F', 'D', 'B', 'C'), ('H', 'G', 'A', 'E', 'F', 'D', 'C', 'B'), ('H', 'G', 'A', 'F', 'B', 'C', 'D', 'E'), ('H', 'G', 'A', 'F', 'B', 'C', 'E', '
C', 'B', 'D'), ('H', 'G', 'A', 'F', 'E', 'C', 'D', 'B'), ('H', 'G', 'A', 'F', 'E', 'D', 'B', 'C'), ('H', 'G', 'A', 'F', 'E', 'D', 'C', 'B'), ('H', 'G', 'B', 'A', 'C', 'D', 'E', 'F'), ('H', 'G', 'B', 'A', 'C', 'D', 'F', 'E'), ('H', 'G',
'H', 'G', 'B', 'A', 'F', 'D', 'E', 'C'), ('H', 'G', 'B', 'A', 'F', 'E', 'C', 'D'), ('H', 'G', 'B', 'A', 'F', 'E', 'D', 'C'), ('H', 'G', 'B', 'C', 'A', 'D', 'E', 'F'), ('H', 'G', 'B', 'C', 'A', 'D', 'F', 'E'), ('H', 'G', 'B', 'C', 'A', '
C', 'F', 'D', 'E', 'A'), ('H', 'G', 'B', 'C', 'F', 'E', 'A', 'D'), ('H', 'G', 'B', 'C', 'F', 'E', 'D', 'A'), ('H', 'G', 'B', 'D', 'A', 'C', 'E', 'F'), ('H', 'G', 'B', 'D', 'A', 'C', 'F', 'E'), ('H', 'G', 'B', 'D', 'A', 'E', 'C', 'F'), (
', 'A'), ('H', 'G', 'B', 'D', 'F', 'E', 'A', 'C'), ('H', 'G', 'B', 'D', 'F', 'E', 'C', 'A'), ('H', 'G', 'B', 'E', 'A', 'C', 'D', 'F'), ('H', 'G', 'B', 'E', 'A', 'C', 'F', 'D'), ('H', 'G', 'B', 'E', 'A', 'D', 'C', 'F'), ('H', 'G', 'B', '
G', 'B', 'E', 'F', 'D', 'A', 'C'), ('H', 'G', 'B', 'E', 'F', 'D', 'C', 'A'), ('H', 'G', 'B', 'F', 'A', 'C', 'D', 'E'), ('H', 'G', 'B', 'F', 'A', 'C', 'E', 'D'), ('H', 'G', 'B', 'F', 'A', 'D', 'C', 'E'), ('H', 'G', 'B', 'F', 'A', 'D', 'E
', 'D', 'A', 'C'), ('H', 'G', 'B', 'F', 'E', 'D', 'C', 'A'), ('H', 'G', 'C', 'A', 'B', 'D', 'E', 'F'), ('H', 'G', 'C', 'A', 'B', 'D', 'F', 'E'), ('H', 'G', 'C', 'A', 'B', 'E', 'D', 'F'), ('H', 'G', 'C', 'A', 'B', 'E', 'F', 'D'), ('H', '
), ('H', 'G', 'C', 'A', 'F', 'E', 'D', 'B'), ('H', 'G', 'C', 'B', 'A', 'D', 'E', 'F'), ('H', 'G', 'C', 'B', 'A', 'D', 'F', 'E'), ('H', 'G', 'C', 'B', 'A', 'E', 'D', 'F'), ('H', 'G', 'C', 'B', 'A', 'E', 'F', 'D'), ('H', 'G', 'C', 'B', 'A
', 'B', 'F', 'E', 'D', 'A'), ('H', 'G', 'C', 'D', 'A', 'B', 'E', 'F'), ('H', 'G', 'C', 'D', 'A', 'B', 'F', 'E'), ('H', 'G', 'C', 'D', 'A', 'E', 'B', 'F'), ('H', 'G', 'C', 'D', 'A', 'E', 'F', 'B'), ('H', 'G', 'C', 'D', 'A', 'F', 'B', 'E'
, 'B', 'A'), ('H', 'G', 'C', 'E', 'A', 'B', 'D', 'F'), ('H', 'G', 'C', 'E', 'A', 'B', 'F', 'D'), ('H', 'G', 'C', 'E', 'A', 'D', 'B', 'F'), ('H', 'G', 'C', 'E', 'A', 'D', 'F', 'B'), ('H', 'G', 'C', 'E', 'A', 'F', 'B', 'D'), ('H', 'G', 'C
', 'G', 'C', 'F', 'A', 'B', 'D', 'E'), ('H', 'G', 'C', 'F', 'A', 'B', 'E', 'D'), ('H', 'G', 'C', 'F', 'A', 'D', 'B', 'E'), ('H', 'G', 'C', 'F', 'A', 'D', 'E', 'B'), ('H', 'G', 'C', 'F', 'A', 'E', 'B', 'D'), ('H', 'G', 'C', 'F', 'A', 'E'
, 'B', 'C', 'E', 'F'), ('H', 'G', 'D', 'A', 'B', 'C', 'F', 'E'), ('H', 'G', 'D', 'A', 'B', 'E', 'C', 'F'), ('H', 'G', 'D', 'A', 'B', 'E', 'F', 'C'), ('H', 'G', 'D', 'A', 'B', 'F', 'C', 'E'), ('H', 'G', 'D', 'A', 'B', 'F', 'E', 'C'), ('H
'F'), ('H', 'G', 'D', 'B', 'A', 'C', 'F', 'E'), ('H', 'G', 'D', 'B', 'A', 'E', 'C', 'F'), ('H', 'G', 'D', 'B', 'A', 'E', 'F', 'C'), ('H', 'G', 'D', 'B', 'A', 'F', 'C', 'E'), ('H', 'G', 'D', 'B', 'A', 'F', 'E', 'C'), ('H', 'G', 'D', 'B',
'D', 'C', 'A', 'B', 'F', 'E'), ('H', 'G', 'D', 'C', 'A', 'E', 'B', 'F'), ('H', 'G', 'D', 'C', 'A', 'E', 'F', 'B'), ('H', 'G', 'D', 'C', 'A', 'F', 'B', 'E'), ('H', 'G', 'D', 'C', 'A', 'F', 'E', 'B'), ('H', 'G', 'D', 'C', 'B', 'A', 'E', '
B', 'F', 'C'), ('H', 'G', 'D', 'E', 'A', 'C', 'B', 'F'), ('H', 'G', 'D', 'E', 'A', 'C', 'F', 'B'), ('H', 'G', 'D', 'E', 'A', 'F', 'B', 'C'), ('H', 'G', 'D', 'E', 'A', 'F', 'C', 'B'), ('H', 'G', 'D', 'E', 'B', 'A', 'C', 'F'), ('H', 'G',
'H', 'G', 'D', 'F', 'A', 'C', 'B', 'E'), ('H', 'G', 'D', 'F', 'A', 'C', 'E', 'B'), ('H', 'G', 'D', 'F', 'A', 'E', 'B', 'C'), ('H', 'G', 'D', 'F', 'A', 'E', 'C', 'B'), ('H', 'G', 'D', 'F', 'B', 'A', 'C', 'E'), ('H', 'G', 'D', 'F', 'B', '
A', 'B', 'D', 'C', 'F'), ('H', 'G', 'E', 'A', 'B', 'D', 'F', 'C'), ('H', 'G', 'E', 'A', 'B', 'F', 'C', 'D'), ('H', 'G', 'E', 'A', 'B', 'F', 'D', 'C'), ('H', 'G', 'E', 'A', 'C', 'B', 'D', 'F'), ('H', 'G', 'E', 'A', 'C', 'B', 'F', 'D'), (
', 'F'), ('H', 'G', 'E', 'B', 'A', 'D', 'F', 'C'), ('H', 'G', 'E', 'B', 'A', 'F', 'C', 'D'), ('H', 'G', 'E', 'B', 'A', 'F', 'D', 'C'), ('H', 'G', 'E', 'B', 'C', 'A', 'D', 'F'), ('H', 'G', 'E', 'B', 'C', 'A', 'F', 'D'), ('H', 'G', 'E', '
G', 'E', 'C', 'A', 'D', 'F', 'B'), ('H', 'G', 'E', 'C', 'A', 'F', 'B', 'D'), ('H', 'G', 'E', 'C', 'A', 'F', 'D', 'B'), ('H', 'G', 'E', 'C', 'B', 'A', 'D', 'F'), ('H', 'G', 'E', 'C', 'B', 'A', 'F', 'D'), ('H', 'G', 'E', 'C', 'B', 'D', 'A
', 'C', 'F', 'B'), ('H', 'G', 'E', 'D', 'A', 'F', 'B', 'C'), ('H', 'G', 'E', 'D', 'A', 'F', 'C', 'B'), ('H', 'G', 'E', 'D', 'B', 'A', 'C', 'F'), ('H', 'G', 'E', 'D', 'B', 'A', 'F', 'C'), ('H', 'G', 'E', 'D', 'B', 'C', 'A', 'F'), ('H', '
), ('H', 'G', 'E', 'F', 'A', 'D', 'B', 'C'), ('H', 'G', 'E', 'F', 'A', 'D', 'C', 'B'), ('H', 'G', 'E', 'F', 'B', 'A', 'C', 'D'), ('H', 'G', 'E', 'F', 'B', 'A', 'D', 'C'), ('H', 'G', 'E', 'F', 'B', 'C', 'A', 'D'), ('H', 'G', 'E', 'F', 'B
', 'A', 'B', 'E', 'C', 'D'), ('H', 'G', 'F', 'A', 'B', 'E', 'D', 'C'), ('H', 'G', 'F', 'A', 'C', 'B', 'D', 'E'), ('H', 'G', 'F', 'A', 'C', 'B', 'E', 'D'), ('H', 'G', 'F', 'A', 'C', 'D', 'B', 'E'), ('H', 'G', 'F', 'A', 'C', 'D', 'E', 'B'
, 'C', 'D'), ('H', 'G', 'F', 'B', 'A', 'E', 'D', 'C'), ('H', 'G', 'F', 'B', 'C', 'A', 'D', 'E'), ('H', 'G', 'F', 'B', 'C', 'A', 'E', 'D'), ('H', 'G', 'F', 'B', 'C', 'D', 'A', 'E'), ('H', 'G', 'F', 'B', 'C', 'D', 'E', 'A'), ('H', 'G', 'F
', 'G', 'F', 'C', 'A', 'E', 'D', 'B'), ('H', 'G', 'F', 'C', 'B', 'A', 'D', 'E'), ('H', 'G', 'F', 'C', 'B', 'A', 'E', 'D'), ('H', 'G', 'F', 'C', 'B', 'D', 'A', 'E'), ('H', 'G', 'F', 'C', 'B', 'D', 'E', 'A'), ('H', 'G', 'F', 'C', 'B', 'E'
, 'A', 'E', 'C', 'B'), ('H', 'G', 'F', 'D', 'B', 'A', 'C', 'E'), ('H', 'G', 'F', 'D', 'B', 'A', 'E', 'C'), ('H', 'G', 'F', 'D', 'B', 'C', 'A', 'E'), ('H', 'G', 'F', 'D', 'B', 'C', 'E', 'A'), ('H', 'G', 'F', 'D', 'B', 'E', 'A', 'C'), ('H
'B'), ('H', 'G', 'F', 'E', 'B', 'A', 'C', 'D'), ('H', 'G', 'F', 'E', 'B', 'A', 'D', 'C'), ('H', 'G', 'F', 'E', 'B', 'C', 'A', 'D'), ('H', 'G', 'F', 'E', 'B', 'C', 'D', 'A'), ('H', 'G', 'F', 'E', 'B', 'D', 'A', 'C'), ('H', 'G', 'F', 'E',
for A: 0.0
for B: 0.0
for C: 0.0
for D: 2.857142857142857
for E: 11.428571428571429
for F: 28.57142857142857
for G: 57.14285714285714
for H: 0.0
```

Appendix 7

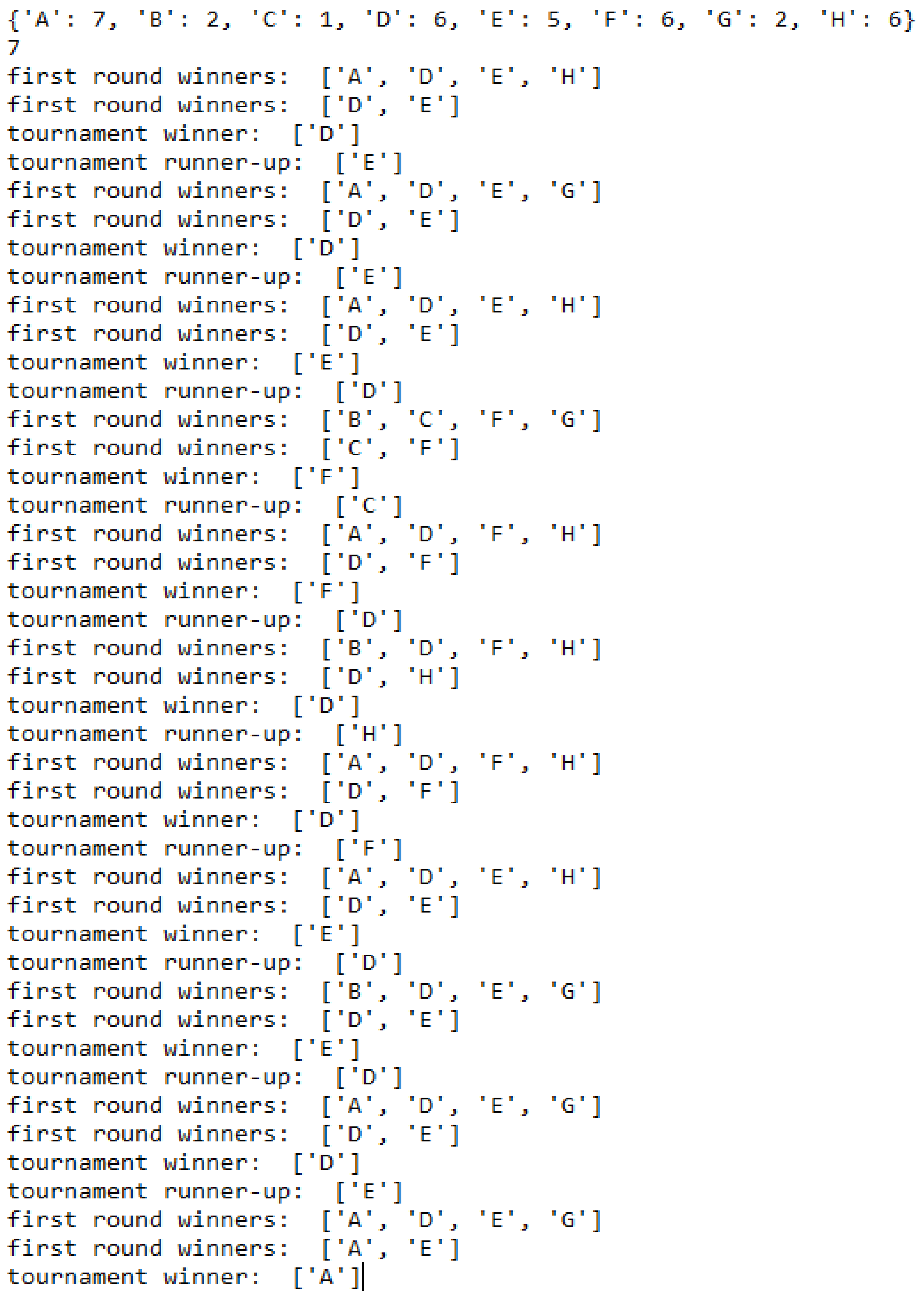


```
from random import randint
import itertools
from math import factorial
p= ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

num= len(p)

#perm= list(itertools.permutations(p))
#eleman = {str(randint(1,num)) for i in p}


dictVal = {x: randint(1,num)  for x in p}

print(dictVal)
print(dictVal["A"])



def victors(para1,para2): #p,dictVal
  route=[]
  i=0
  while i in range(len(para1)-1):
    a= [para1[i], para1[i+1]]
    b= [para2[a[0]], para2[a[1]]]
    total=int(b[0])+ int(b[1])
    rnd= randint(1,total)
    if (rnd in range(1,int(b[0])+1) ):
      vct= a[0]
    else:
      vct= a[1]
    route.append(vct)
    i+=2
  return route

valA=0
valB=0
valC=0
valD=0
valE=0
valF=0
valG=0
valH=0
valA1=0
valB1=0
valC1=0
valD1=0
valE1=0
valF1=0
valG1=0
valH1=0
```

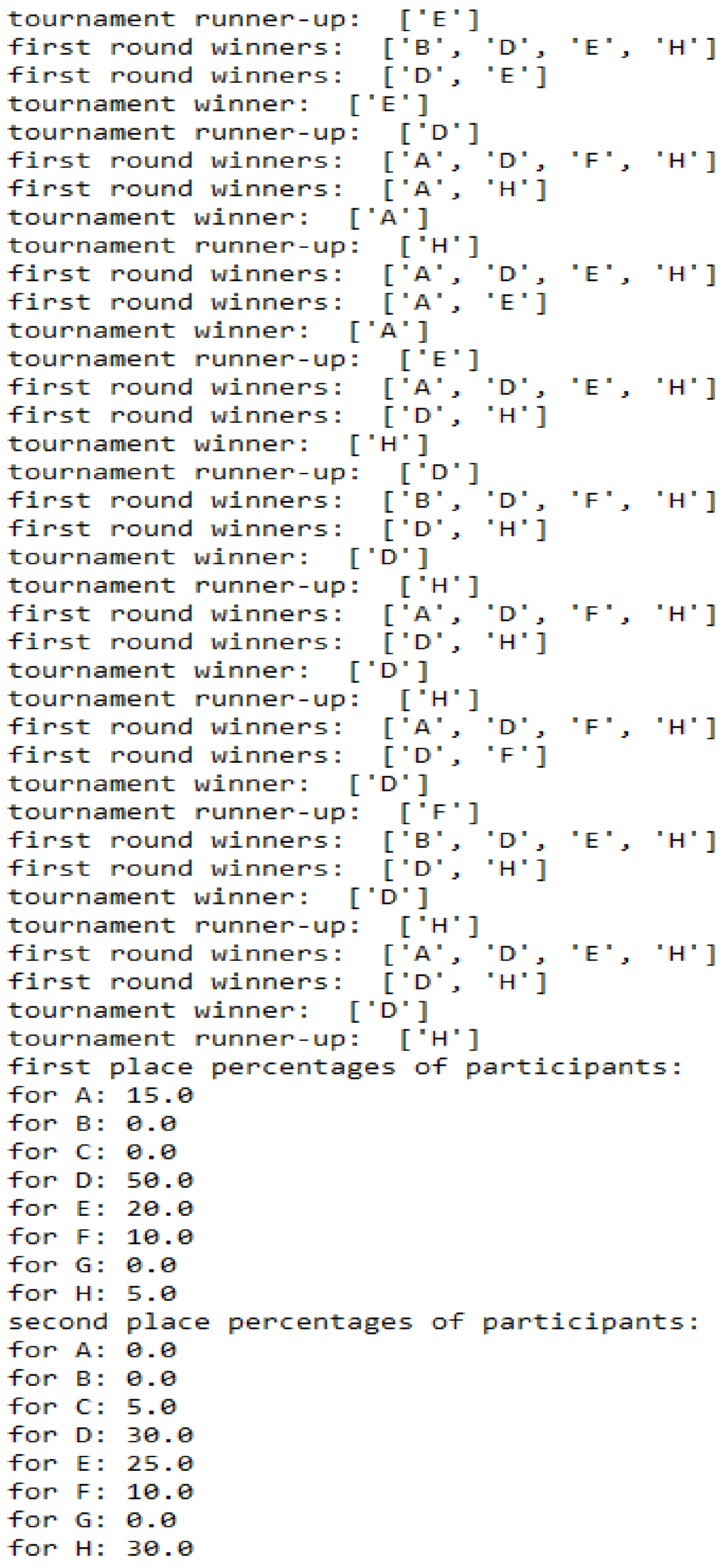


```
RANGE= 20

for a in range(RANGE):
  route1= victors(p,dictVal)
  route2= victors(route1,dictVal)
  winner= victors(route2,dictVal)
  second = route2.copy()
  second.remove(winner[0])
  print("first round winners: ",route1)
  print("second round winners: ",route2)
  print("tournament winner: ",winner)
  print("tournament runner-up: ",second)

  if (second== ['A']):
    valA+=1
  elif (second== ['B']):
    valB+=1
  elif (second== ['C']):
    valC+=1
  elif (second== ['D']):
    valD+=1
  elif (second== ['E']):
    valE+=1
  elif (second== ['F']):
    valF+=1
  elif (second== ['G']):
    valG+=1
  elif (second== ['H']):
    valH+=1
  if (winner== ['A']):
    valA1+=1
  elif (winner== ['B']):
    valB1+=1
  elif (winner== ['C']):
    valC1+=1
  elif (winner== ['D']):
    valD1+=1
  elif (winner== ['E']):
    valE1+=1
  elif (winner== ['F']):
    valF1+=1
  elif (winner== ['G']):
    valG1+=1
  elif (winner== ['H']):
    valH1+=1



print("first place percentages of participants: " )
print("for A:" , valA1/RANGE *100 )
print("for B:" , valB1/RANGE *100 )
print("for C:" , valC1/RANGE *100 )
print("for D:" , valD1/RANGE *100 )
print("for E:" , valE1/RANGE *100 )
print("for F:" , valF1/RANGE *100 )
print("for G:" , valG1/RANGE *100 )
print("for H:" , valH1/RANGE *100 )

print("second place percentages of participants: " )
print("for A:" , valA/RANGE  *100)
print("for B:" , valB/RANGE  *100)
print("for C:" , valC/RANGE  *100)
print("for D:" , valD/RANGE  *100)
print("for E:" , valE/RANGE  *100)
print("for F:" , valF/RANGE  *100)
print("for G:" , valG/RANGE  *100)
print("for H:" , valH/RANGE  *100)
```

Appendix 7 Output



```
{'A': 7, 'B': 2, 'C': 1, 'D': 6, 'E': 5, 'F': 6, 'G': 2, 'H': 6}
7
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['D', 'E']
tournament winner:  ['D']
tournament runner-up:  ['E']
first round winners:  ['A', 'D', 'E', 'G']
first round winners:  ['D', 'E']
tournament winner:  ['D']
tournament runner-up:  ['E']
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['D', 'E']
tournament winner:  ['E']
tournament runner-up:  ['D']
first round winners:  ['B', 'C', 'F', 'G']
first round winners:  ['C', 'F']
tournament winner:  ['F']
tournament runner-up:  ['C']
first round winners:  ['A', 'D', 'F', 'H']
first round winners:  ['D', 'F']
tournament winner:  ['F']
tournament runner-up:  ['D']
first round winners:  ['B', 'D', 'F', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['D']
tournament runner-up:  ['H']
first round winners:  ['A', 'D', 'F', 'H']
first round winners:  ['D', 'F']
tournament winner:  ['D']
tournament runner-up:  ['F']
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['D', 'E']
tournament winner:  ['E']
tournament runner-up:  ['D']
first round winners:  ['B', 'D', 'E', 'G']
first round winners:  ['D', 'E']
tournament winner:  ['E']
tournament runner-up:  ['D']
first round winners:  ['A', 'D', 'E', 'G']
first round winners:  ['D', 'E']
tournament winner:  ['D']
tournament runner-up:  ['E']
first round winners:  ['A', 'D', 'E', 'G']
first round winners:  ['A', 'E']
tournament winner:  ['A']
```



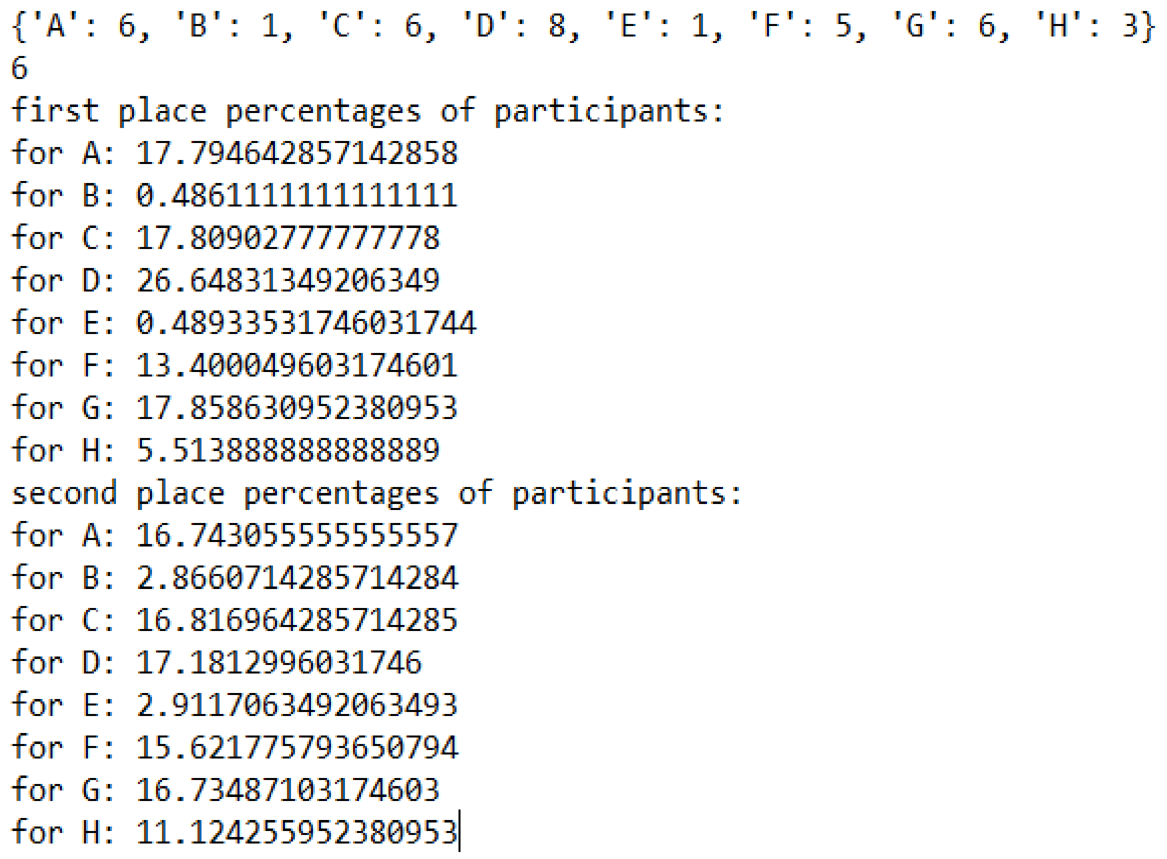
```
tournament runner-up:  ['E']
first round winners:  ['B', 'D', 'E', 'H']
first round winners:  ['D', 'E']
tournament winner:  ['E']
tournament runner-up:  ['D']
first round winners:  ['A', 'D', 'F', 'H']
first round winners:  ['A', 'H']
tournament winner:  ['A']
tournament runner-up:  ['H']
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['A', 'E']
tournament winner:  ['A']
tournament runner-up:  ['E']
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['H']
tournament runner-up:  ['D']
first round winners:  ['B', 'D', 'F', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['D']
tournament runner-up:  ['H']
first round winners:  ['A', 'D', 'F', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['D']
tournament runner-up:  ['H']
first round winners:  ['A', 'D', 'F', 'H']
first round winners:  ['D', 'F']
tournament winner:  ['D']
tournament runner-up:  ['F']
first round winners:  ['B', 'D', 'E', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['D']
tournament runner-up:  ['H']
first round winners:  ['A', 'D', 'E', 'H']
first round winners:  ['D', 'H']
tournament winner:  ['D']
tournament runner-up:  ['H']
first place percentages of participants:
for A: 15.0
for B: 0.0
for C: 0.0
for D: 50.0
for E: 20.0
for F: 10.0
for G: 0.0
for H: 5.0
second place percentages of participants:
for A: 0.0
for B: 0.0
for C: 5.0
for D: 30.0
for E: 25.0
for F: 10.0
for G: 0.0
for H: 30.0
```

Appendix 8

```
from random import randint
import itertools
import math
p= ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

num= len(p)

perm= list(itertools.permutations(p))
#eleman = {str(randint(1,num)) for i in p}


dictVal = {x: randint(1,num)  for x in p}

print(dictVal)
print(dictVal["A"])



def victors(para1,para2): #p,dictVal
  route=[]
  i=0
  while i in range(len(para1)-1):
    a= [para1[i], para1[i+1]]
    b= [para2[a[0]], para2[a[1]]]
    total=int(b[0])+ int(b[1])
    rnd= randint(1,total)
    if (rnd in range(1,int(b[0])+1) ):
      vct= a[0]
    else:
      vct= a[1]
    route.append(vct)
    i+=2
  return route

valA=0
valB=0
valC=0
valD=0
valE=0
valF=0
valG=0
valH=0
valA1=0
valB1=0
valC1=0
valD1=0
valE1=0
valF1=0
valG1=0
valH1=0
```

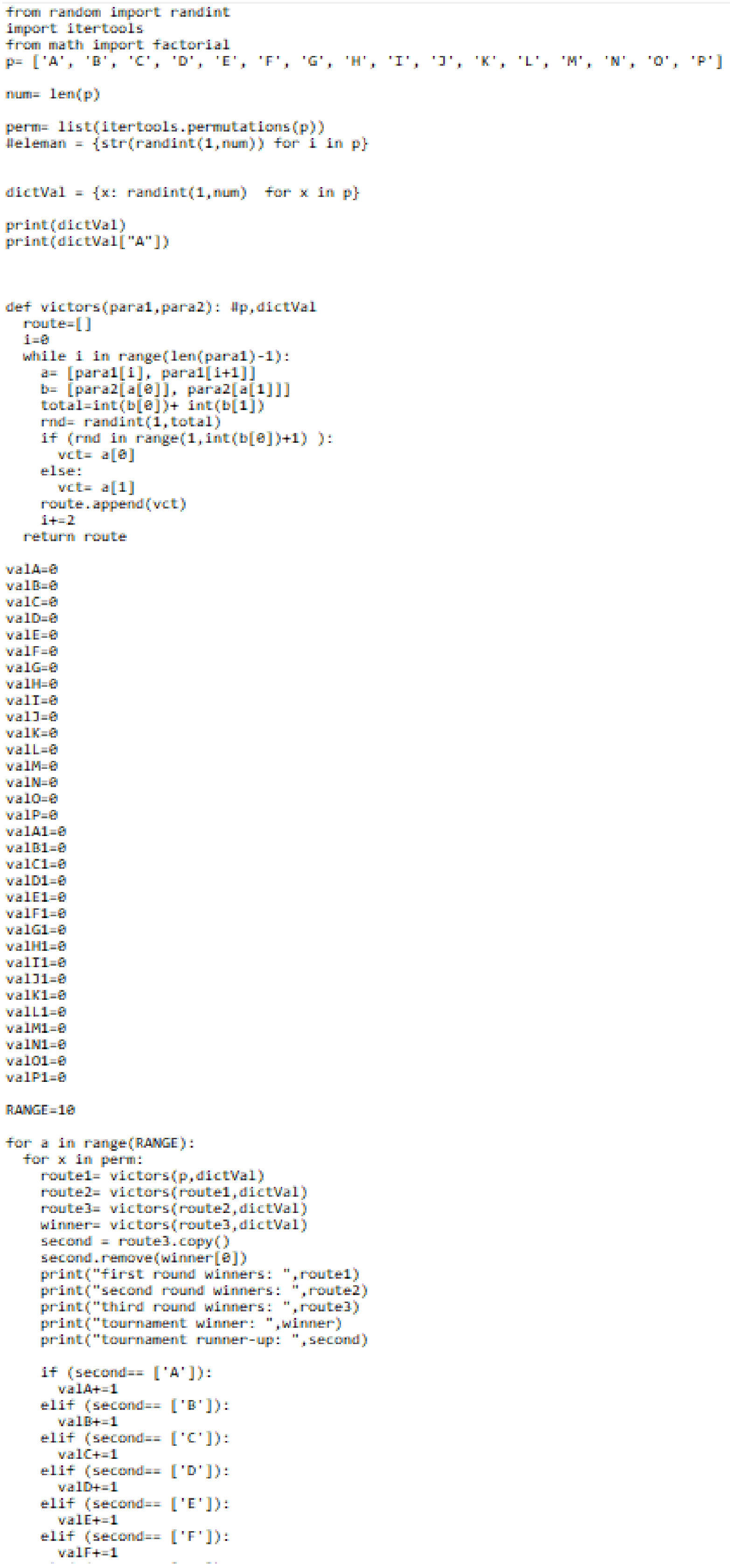
```
RANGE=10
for a in range(RANGE):
  for x in perm:
    route1= victors(x,dictVal)
    route2= victors(route1,dictVal)
    winner= victors(route2,dictVal)
    second = route2.copy()
    second.remove(winner[0])
  # print("first round winners: ",route1)
  # print("first round winners: ",route2)
  # print("tournament winner: ",winner)
  # print("tournament runner-up: ",second)

    if (second== ['A']):
      valA+=1
    elif (second== ['B']):
      valB+=1
    elif (second== ['C']):
      valC+=1
    elif (second== ['D']):
      valD+=1
    elif (second== ['E']):
      valE+=1
    elif (second== ['F']):
      valF+=1
    elif (second== ['G']):
      valG+=1
    elif (second== ['H']):
      valH+=1
    if (winner== ['A']):
      valA1+=1
    elif (winner== ['B']):
      valB1+=1
    elif (winner== ['C']):
      valC1+=1
    elif (winner== ['D']):
      valD1+=1
    elif (winner== ['E']):
      valE1+=1
    elif (winner== ['F']):
      valF1+=1
    elif (winner== ['G']):
      valG1+=1
    elif (winner== ['H']):
      valH1+=1

print("first place percentages of participants: " )
print("for A:" , valA1/(RANGE*math.factorial(len(p)))*100)
print("for B:" , valB1/(RANGE*math.factorial(len(p)))*100)
print("for C:" , valC1/(RANGE*math.factorial(len(p)))*100)
print("for D:" , valD1/(RANGE*math.factorial(len(p)))*100)
print("for E:" , valE1/(RANGE*math.factorial(len(p)))*100)
print("for F:" , valF1/(RANGE*math.factorial(len(p)))*100)
print("for G:" , valG1/(RANGE*math.factorial(len(p)))*100)
print("for H:" , valH1/(RANGE*math.factorial(len(p)))*100)


print("second place percentages of participants: " )
print("for A:" , valA/(RANGE*math.factorial(len(p)))*100)
print("for B:" , valB/(RANGE*math.factorial(len(p)))*100)
print("for C:" , valC/(RANGE*math.factorial(len(p)))*100)
print("for D:" , valD/(RANGE*math.factorial(len(p)))*100)
print("for E:" , valE/(RANGE*math.factorial(len(p)))*100)
print("for F:" , valF/(RANGE*math.factorial(len(p)))*100)
print("for G:" , valG/(RANGE*math.factorial(len(p)))*100)
print("for H:" , valH/(RANGE*math.factorial(len(p)))*100)
```

Appendix 8 Output

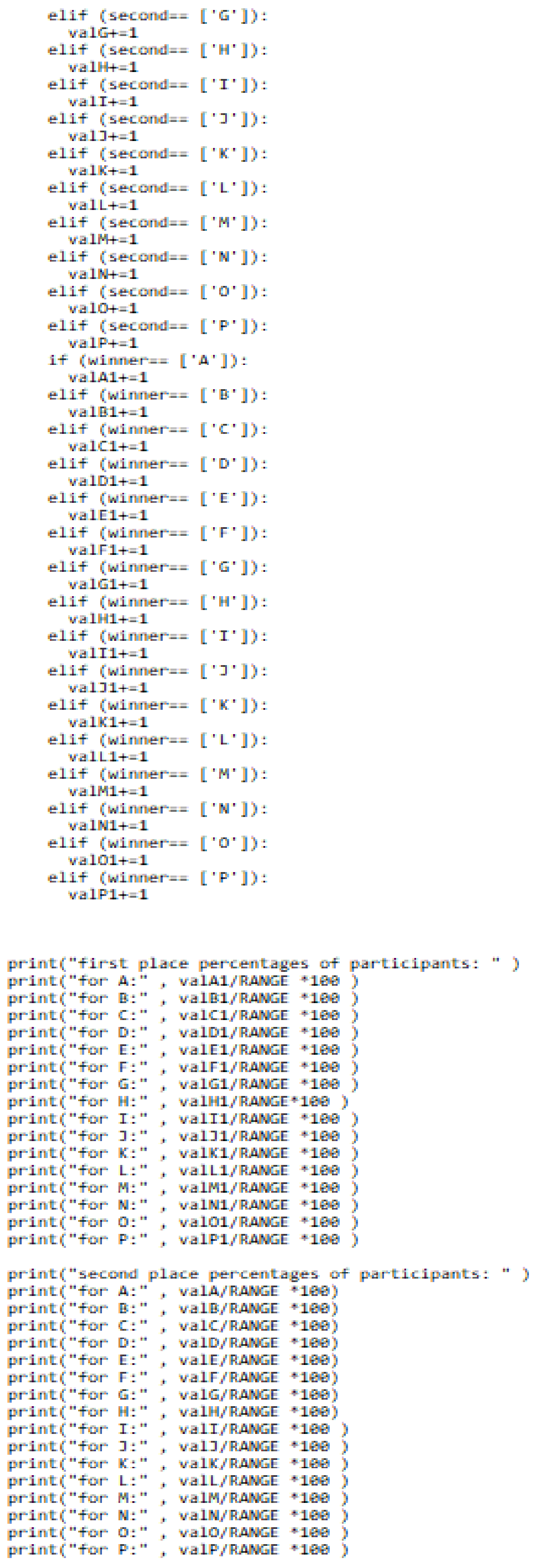


```
{'A': 6, 'B': 1, 'C': 6, 'D': 8, 'E': 1, 'F': 5, 'G': 6, 'H': 3}
6
first place percentages of participants:
for A: 17.794642857142858
for B: 0.4861111111111111
for C: 17.809027777777778
for D: 26.64831349206349
for E: 0.48933531746031744
for F: 13.400049603174601
for G: 17.858630952380953
for H: 5.513888888888889
second place percentages of participants:
for A: 16.743055555555557
for B: 2.8660714285714284
for C: 16.816964285714285
for D: 17.1812996031746
for E: 2.9117063492063493
for F: 15.621775793650794
for G: 16.73487103174603
for H: 11.124255952380953
```

Appendix 9



```
from random import randint
import itertools
from math import factorial
p= ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P']

num= len(p)

perm= list(itertools.permutations(p))
#eleman = {str(randint(1,num)) for i in p}


dictVal = {x: randint(1,num)  for x in p}

print(dictVal)
print(dictVal["A"])



def victors(para1,para2): #p,dictVal
  route=[]
  i=0
  while i in range(len(para1)-1):
    a= [para1[i], para1[i+1]]
    b= [para2[a[0]], para2[a[1]]]
    total=int(b[0])+ int(b[1])
    rnd= randint(1,total)
    if (rnd in range(1,int(b[0])+1) ):
      vct= a[0]
    else:
      vct= a[1]
    route.append(vct)
    i+=2
  return route

valA=0
valB=0
valC=0
valD=0
valE=0
valF=0
valG=0
valH=0
valI=0
valJ=0
valK=0
valL=0
valM=0
valN=0
valO=0
valP=0
valA1=0
valB1=0
valC1=0
valD1=0
valE1=0
valF1=0
valG1=0
valH1=0
valI1=0
valJ1=0
valK1=0
valL1=0
valM1=0
valN1=0
valO1=0
valP1=0

RANGE=10

for a in range(RANGE):
  for x in perm:
    route1= victors(p,dictVal)
    route2= victors(route1,dictVal)
    route3= victors(route2,dictVal)
    winner= victors(route3,dictVal)
    second = route3.copy()
    second.remove(winner[0])
    print("first round winners: ",route1)
    print("second round winners: ",route2)
    print("third round winners: ",route3)
    print("tournament winner: ",winner)
    print("tournament runner-up: ",second)

    if (second== ['A']):
      valA+=1
    elif (second== ['B']):
      valB+=1
    elif (second== ['C']):
      valC+=1
    elif (second== ['D']):
      valD+=1
    elif (second== ['E']):
      valE+=1
    elif (second== ['F']):
      valF+=1
```



```
    elif (second== ['G']):
      valG+=1
    elif (second== ['H']):
      valH+=1
    elif (second== ['I']):
      valI+=1
    elif (second== ['J']):
      valJ+=1
    elif (second== ['K']):
      valK+=1
    elif (second== ['L']):
      valL+=1
    elif (second== ['M']):
      valM+=1
    elif (second== ['N']):
      valN+=1
    elif (second== ['O']):
      valO+=1
    elif (second== ['P']):
      valP+=1
    if (winner== ['A']):
      valA1+=1
    elif (winner== ['B']):
      valB1+=1
    elif (winner== ['C']):
      valC1+=1
    elif (winner== ['D']):
      valD1+=1
    elif (winner== ['E']):
      valE1+=1
    elif (winner== ['F']):
      valF1+=1
    elif (winner== ['G']):
      valG1+=1
    elif (winner== ['H']):
      valH1+=1
    elif (winner== ['I']):
      valI1+=1
    elif (winner== ['J']):
      valJ1+=1
    elif (winner== ['K']):
      valK1+=1
    elif (winner== ['L']):
      valL1+=1
    elif (winner== ['M']):
      valM1+=1
    elif (winner== ['N']):
      valN1+=1
    elif (winner== ['O']):
      valO1+=1
    elif (winner== ['P']):
      valP1+=1


print("first place percentages of participants: " )
print("for A:" , valA1/RANGE *100 )
print("for B:" , valB1/RANGE *100 )
print("for C:" , valC1/RANGE *100 )
print("for D:" , valD1/RANGE *100 )
print("for E:" , valE1/RANGE *100 )
print("for F:" , valF1/RANGE *100 )
print("for G:" , valG1/RANGE *100 )
print("for H:" , valH1/RANGE*100 )
print("for I:" , valI1/RANGE *100 )
print("for J:" , valJ1/RANGE *100 )
print("for K:" , valK1/RANGE *100 )
print("for L:" , valL1/RANGE *100 )
print("for M:" , valM1/RANGE *100 )
print("for N:" , valN1/RANGE *100 )
print("for O:" , valO1/RANGE *100 )
print("for P:" , valP1/RANGE *100 )

print("second place percentages of participants: " )
print("for A:" , valA/RANGE *100)
print("for B:" , valB/RANGE *100)
print("for C:" , valC/RANGE *100)
print("for D:" , valD/RANGE *100)
print("for E:" , valE/RANGE *100)
print("for F:" , valF/RANGE *100)
print("for G:" , valG/RANGE *100)
print("for H:" , valH/RANGE *100)
print("for I:" , valI/RANGE *100 )
print("for J:" , valJ/RANGE *100 )
print("for K:" , valK/RANGE *100 )
print("for L:" , valL/RANGE *100 )
print("for M:" , valM/RANGE *100 )
print("for N:" , valN/RANGE *100 )
print("for O:" , valO/RANGE *100 )
print("for P:" , valP/RANGE *100 )
```