CSE 591: Foundations of Algorithms
Programming Assignment # 2
Rachita Gupta: 1209247724

1. Introduction

Organize a tournament involving n players. Each competitor must play exactly once against each of its opponents. Each player must play exactly once each day, with possible exception of a single day when he does not play at all.
For any integer n > 1 give an algorithm to construct a timetable allowing the tournament to be finished in (n – 1) days if n is even, or in n days if n is odd

2. Algorithm

Tournament algorithm works as follows. It arranges all the teams in an order and pair all the teams in such a way that the first team plays with the last, second team plays with the second last and so on. Then it rotates all the teams in the above arrangement such that all the teams moves 1 step ahead and the last team comes at first position. This course is repeated until we get the original order back.
There are 2 special cases when number of teams is odd and when it is even. In case of odd there will always be team left in the center of the arrangement. For each round that team will not play. In case of even number of teams, every time we rotate n-1 number of teams. In this case also the team in the center will be left. So, for every round this team is paired with the last nth team.

N=5	N=6
1,2, (3) , 4,5 : 1-5, 2-4	**1,2,3,4,5,6: 1-6,2-5,3-4**
5,1, (2) ,3,4: 5-4, 1-3	**5,1,2,3,4,6: 5-6,1-4,2-3**
4,5, (1) ,2,3: 4-3, 5-2	**4,5,1,2,3,6: 4-6,5-3,1-2**
3,4, (5) ,1,2: 3-2, 4-1	**3,4,5,1,2,6: 3-6,4-2,5-1**
2,3, (4) ,5,1: 2:1, 3-5	**2,3,4,5,1,6: 2-6,3-1,4-5**

3. Pseudo code

Input: n competitors
A[1,2,…n]
Play(n)
for days ← 1 to n
for j ← 1 to n/2
A[j] plays with A[n-j-1]
rotating A[j] in one direction such that it becomes A[j+1,j+2,…,n,j]
end

4. Proof of correctness

In the above algorithm, for even number of players the tournament lasts n-1 games and for odd players we add I dummy team. So for odd players also it lasts n-1 days.
For each day we iterate through the players till n/2 to pair them with the other half of the players. Therefore there are n/2 games in a day.
After generating the schedule for a day, the algorithm rotates the players. After rotation each player is paired with a different player. This rotation is done till we don't get the same arrangement. This type of rotation makes sure that there is no identical pairing. Hence, at the end we get a unique schedule where each player plays with the other only once.

Let N=3

1. For day=1 the schedule will be 1-3 and player number 2 will not play. So 3/2=1 game
2. For day=2 the schedule will be 3-2 and player 1 does not play. So again 1 match
3. For day=3 schedule will be 1-2 and player 3 does not play. 1 match

Then the algorithm ends. At the end each player plays with all other once and there are n/2 games each day. N being odd the tournament ends in n days

Let N=4

1. For day=1 the schedule will be 1-4, 2-3. So 4/2=2 game
2. For day=2 the schedule will be 1-3, 2-4. So again 2 match
3. For day=3 schedule will be 1-2, 3-4. 1 match

Then the algorithm ends. At the end each player plays with all other once and there are n/2 games. N being even the tournament ends in n-1 days.

For N=n (odd)

1. For day=1, schedule will be 1-n, 2-(n-1), 3-(n-2).. and player number n/2+1 doesn't play. So n/2 games.
2. For day=2 schedule will be n-(n-1), 1-(n-2), 2-(n-3).. and player number n/2 does not play.

.

.
.
n. For day=n, 2-1,3-n,4-(n-1)… and 1 player does not play.

For N=n+1 (even)

1. For day=1, schedule will be 1-n, 2-(n-1), 3-(n-2). So n/2 games.
2. For day=2 schedule will be (n-1)-n, 1-(n-2), 2-(n-3)…
3. For day=3 schedule will be (n-2)-n, (n-1)-(n-3),... n/2 games

.
.
n. For day=n, 2-1, 3-n,4-(n-1)… and 1 player does not play.

For all values of n it is seen that this algorithm satisfies the requirements. Hence this algorithm is correct.

5. Complexity

Using above algorithm, if there are n competitors then there will n/2(n-1) games.
There are 2 cases, one when n is odd and when it is even. For even n there will be (n-1) rounds where n/2 games can be played concurrently. But if n is odd there will be n rounds where (n-1)/2 games are played concurrently.

Time complexity of above algorithm is $T(n)=O(n^2)$

Space complexity is O(n)

6. Output

1.

```
C:\Users\rgrac\Desktop\Documents\MS\FOA\PA2>javac Main.java

C:\Users\rgrac\Desktop\Documents\MS\FOA\PA2>java Main
Select an option
1. Input by File
2. Input by Console
2
Enter number of players
5
1:5:4:-:2:1
2:3:-:1:5:4
3:-:5:4:3:2
4:4:3:2:1:-
5:2:1:5:-:3

select (1/0) to continue
1
Enter number of players
6
1:6:5:4:3:2:1
2:4:3:2:1:6:5
3:2:1:5:6:3:4
4:5:4:6:2:1:3
5:3:6:1:5:4:2

select (1/0) to continue
0
```

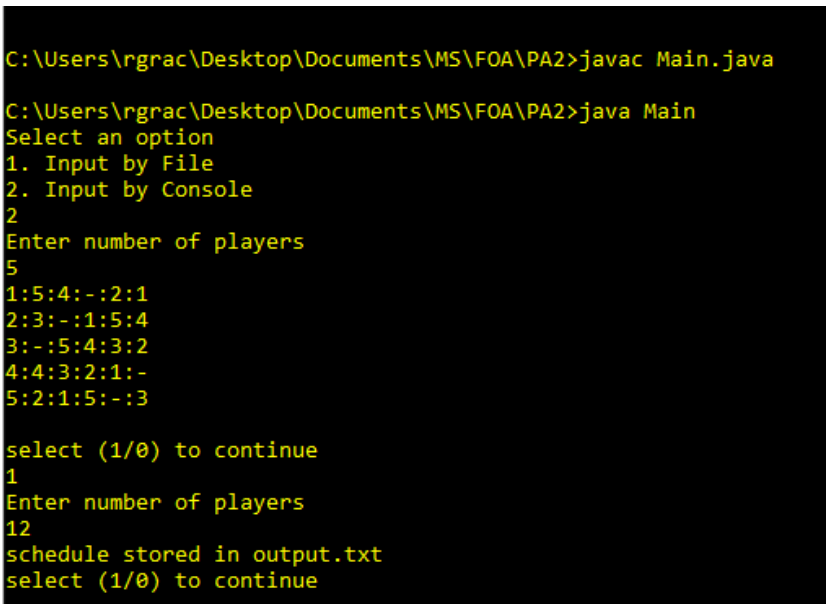
2.

```
C:\Users\rgrac\Desktop\Documents\MS\FOA\PA2>javac Main.java

C:\Users\rgrac\Desktop\Documents\MS\FOA\PA2>java Main
Select an option
1. Input by File
2. Input by Console
1
Enter the name of the input file
input.txt
1:4:3:2:1
2:2:1:4:3
3:3:4:1:2

1:5:4:-:2:1
2:3:-:1:5:4
3:-:5:4:3:2
4:4:3:2:1:-
5:2:1:5:-:3
```

3.



7. Programming Language

The language we have used is Java because we found a lot more useful resources and libraries supporting Java. The second main reason for using the Java programming language was the familiarity of our group with the language. We have had hands on experience with Java during our academic and professional projects making it easier to implement new functionalities and features

8. Conclusion

Using the above algorithm we were successfully able to schedule a tournament for any number of competitors such that everyone plays with the other competitor once.

9. References

[1.] https://en.wikipedia.org/wiki/Round-robin_tournament
[2.] Introduction to Algorithms by Thomas H. Cormen and Charles E. Leiserson