FOA Programming Assignment-2

Algorithm to Solve Problem1:

Following is the algorithm I used to solve the problem.

Let's assume n (number of players) is odd and solve the problem.

The total number of matches is n (n-1)/2. As n is odd there can be at most n-1 matches every day. The fewest number of days required would be n.

This algorithm can be extended to even case as well using the same approach but for now let's assume n is odd. The algorithm for pairing runs as follows:

In each round the element a[i] is paired with the element a[n-i+1] and as this is an odd case we would give a "bye" to the element that is in the middle of the array in each round.

Pseudocode:

```
List all players in array in their increasing order of index
For day=1, 2…(n-1) do
For i=1, 2…n/2 do
Match (a[i], a [n-i+1]);
End
Rotate the players using circular right shift, the element in the last becomes the first element in array.
Eg: {p1, p2…,.pn} becomes {pn,p1,p2,…p(n-1)}
End
```

Sample run of algorithm for 7 players is explained below:

Day 1:

1	2	3	4	5	6	7

Matches: 1-7, 2-6, 3-5, 4 gets a "bye" as it's in the middle.

Day 2:

7	1	2	3	4	5	6

Matches: 7-6, 1-5, 2-4, 3 gets a "bye"

Day 3:

6	7	1	2	3	4	5

Matches: 6-5, 7-4, 1-3, 2 gets a "bye"

Day 4:

5	6	7	1	2	3	4

Matches: 5-4, 6-3, 7-2, 1 gets a “bye”

Day 5:

4	5	6	7	1	2	3

Matches: 4-3, 5-2, 6-1, 7 gets a “bye”

Day 6:

3	4	5	6	7	1	2

Matches: 3-2, 4-1, 5-7, 6 gets a “bye”

Day 7:

2	3	4	5	6	7	1

Matches: 2-1, 3-7, 4-6, 5 gets a “bye”

Even Case:

The Even case is extended from above odd case and matching with the new element we take. Suppose for n=8 we run the algorithm for n=7 and then match the player 8 with the player who got a “bye” in each round. This algorithm is also correct as all the 7 other players are matched with 8 only once as they become the middle element of the array only once in the whole run of the algorithm. As per above odd algorithm, all the other 7 players are also matched once. I would be proving its correctness in the subsequent sections.

<u>Proof of Correctness:</u>

1. **n is odd**

We would prove the algorithm for the odd case and extend it to prove the even case as well. Let n be the number of players and hence the tournament would require n-1 days.

<u>Claim 1:</u> There is a player left out on each day.

As there are odd number of players and a match requires two players, there is a definite chance that one player is left out on every day. We give him a “bye”.

<u>Claim 2</u>: Every player plays with every other player only once.

As per my algorithm, a player gets matched only if he is equidistant with another player from the center.

For e.g.: 1 2 3 4 5 6 7

1-7 gets matched only when 4 which is the mid of the array is equidistant from 1 and 7.Here we take two cases:

<u>Case 1:</u> Two players which have odd number of players between them get matched only when the mid of them is at the mid of the array.

As per my algorithm, 1 and 3 would get matched only when their mid i.e 2 would be at the center of the array.2 would be at the center of array just once in the whole algorithm. Similar is the case with 2 and 6, they would get matched only when their mid i.e 4 is at the center of array and equidistant from each of 2 and 6. The same explanation can be extended to n.

Case 2: Two players which have even number of players between them would not play until the number of players between them become odd.

As per algorithm, match happens only when there is a decisive mid value. If there is an even number of players between two players, they would never play until one of them gets rotated to the start of array, by changing the number of players between them to odd. Once, such a rotation happens, the mid value between the two values in consideration changes and also, the new mid value of the two values would not have crossed the mid of array. This is because of fact that after rotation, the max number of values between two players in consideration can be at most length of array-2.Hence, the mid of the players now with odd number of players also would be at mid of array only once. Hence, the players with even number of players between them also get matched only once.

Eg. 1 2 3 4 5 6 7

1 and 4 gets never matched unless one of them gets rotated.

2. n is even

As per my algorithm, I am running and odd instance for n-1 and matching the team that got a bye with the last player in the array. Even then, all the proof above holds as the n-1 players have confirmed to the claims above. The nth player is always getting matched to the player with a bye and also each player gets a bye only once for the n-1 players. Hence nth player plays n-1 matches with a unique player every time.

This completes the proof of correctness of the algorithm.

Programming language used and reason for doing so:

Java has a robust set of built in functions and is a language which is easy to code than its counterparts like c, cpp. It is platform independent and the same program can be executed on a variety of platforms.

Sample Output:

```
SELECT YOUR CHOICE 1.CONSOLE 2.TEXTFILE 3.EXIT // Input is taken from Console
1
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 YOUR CHOICE 1.CONSOLE 2.TEXTFILE 3.EXIT // Input is taken from Console
1
6
1:5:4:6:2:1:3
2:3:6:1:5:4:2
3:6:5:4:3:2:1
4:4:3:2:1:6:5
5:2:1:5:6:3:4
```

```
//Input is taken from file and only values less than or equal to 10 are
printed to console, values >10 are printed to file output.txt.
SELECT YOUR CHOICE 1.CONSOLE 2.TEXTFILE 3.EXIT
2
1:3:4:1:2
2:4:3:2:1
3:2:1:4:3
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
1:7:6:5:8:3:2:1:4
2:5:4:8:2:1:7:6:3
3:3:8:1:7:6:5:4:2
4:8:7:6:5:4:3:2:1
5:6:5:4:3:2:1:8:7
6:4:3:2:1:7:8:5:6
7:2:1:7:6:8:4:3:5
1:5:4:6:2:1:3
2:3:6:1:5:4:2
3:6:5:4:3:2:1
4:4:3:2:1:6:5
5:2:1:5:6:3:4
1:7:6:5:-:3:2:1
2:5:4:-:2:1:7:6
3:3:-:1:7:6:5:4
4:-:7:6:5:4:3:2
5:6:5:4:3:2:1:-
6:4:3:2:1:7:-:5
7:2:1:7:6:-:4:3
1:3:-:1
2:-:3:2
3:2:1:-
1:2:1
SELECT YOUR CHOICE 1.CONSOLE 2.TEXTFILE 3.EXIT
3
```

Input file contents:

```
4
5
8
6
7
32
11
3
2
```

Output file contents:

```
1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:32:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:16
2:29:28:27:26:25:24:23:22:21:20:19:18:17:16:32:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:15
3:27:26:25:24:23:22:21:20:19:18:17:16:15:32:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:14
4:25:24:23:22:21:20:19:18:17:16:15:14:32:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:13
5:23:22:21:20:19:18:17:16:15:14:13:32:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:12
6:21:20:19:18:17:16:15:14:13:12:32:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:11
```

```
7:19:18:17:16:15:14:13:12:11:32:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:10
8:17:16:15:14:13:12:11:10:32:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:9
9:15:14:13:12:11:10:9:32:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:8
10:13:12:11:10:9:8:32:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:7
11:11:10:9:8:7:32:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:6
12:9:8:7:6:32:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:5
13:7:6:5:32:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:4
14:5:4:32:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:3
15:3:32:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:2
16:32:31:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1
17:30:29:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:32:31
18:28:27:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:32:29:30
19:26:25:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:32:28:27:29
20:24:23:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:32:27:26:25:28
21:22:21:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:32:26:25:24:23:27
22:20:19:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:32:25:24:23:22:21:26
23:18:17:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:32:24:23:22:21:20:19:25
24:16:15:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:32:23:22:21:20:19:18:17:24
25:14:13:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:32:22:21:20:19:18:17:16:15:23
26:12:11:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:32:21:20:19:18:17:16:15:14:13:22
27:10:9:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:32:20:19:18:17:16:15:14:13:12:11:21
28:8:7:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:32:19:18:17:16:15:14:13:12:11:10:9:20
29:6:5:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:32:18:17:16:15:14:13:12:11:10:9:8:7:19
30:4:3:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:32:17:16:15:14:13:12:11:10:9:8:7:6:5:18
31:2:1:31:30:29:28:27:26:25:24:23:22:21:20:19:18:32:16:15:14:13:12:11:10:9:8:7:6:5:4:3:17
################################################################
1:11:10:9:8:7:-:5:4:3:2:1
2:9:8:7:6:-:4:3:2:1:11:10
3:7:6:5:-:3:2:1:11:10:9:8
4:5:4:-:2:1:11:10:9:8:7:6
5:3:-:1:11:10:9:8:7:6:5:4
6:-:11:10:9:8:7:6:5:4:3:2
7:10:9:8:7:6:5:4:3:2:1:-
8:8:7:6:5:4:3:2:1:11:-:9
9:6:5:4:3:2:1:11:10:-:8:7
10:4:3:2:1:11:10:9:-:7:6:5
11:2:1:11:10:9:8:-:6:5:4:3
######################
```