Report of prolog project
Main Predicates :
1- schedule_round(N , Schedule)

this predicate will take one input N the number of players and will return the Schedule for the corresponding round. It will simply create 2 lists one with the first half of players and the one with the second half and will call the predicate schedule_helper (FirstHalf , SecondHalf , Schedule , N) (explained in the helper predicates).
2- schedule_rounds(N, R)

this predicate will take the number of players and will create a list of schedule_round played by these players.

3- tournament(N ,Days , Result)

this predicate will take 2 inputs : N number of players and Days available days for the whole tournament for this number of players and should return the Resulted schedule for arranging this tournament for N players in Days.
This predicate will create a schedule_round for N players and will callPick2 on this schedules after initializing length to 3 and will repeat for all rounds and finally will append all the results.

Another solution for 3

3- tournament(N ,Days , Result) :-
this predicate will take 2 inputs : N number of players and Days available days for the whole tournament for this number of players and should return the Resulted schedule for arranging this tournament for N players in Days.
This predicate will create a schedule_round for N players and will call real_generate on this schedules and will repeat for all rounds and finally will append all the results.

Helper Predicates:
1 – create (Start , List , End)
This predicate will take Start and End (that are the limits of the elements of the List) and will create a list of elements starting from Start to End.
For ex : ?- create(1,X,4).
X = [1, 2, 3, 4] ;
false.
?- create(3,X,5).
X = [3, 4, 5] ;
false.
2- pick (Element , List ,Rest)
This predicate will choose an Element from List and will return the Rest of the List after removing Element.
For ex : ?- pick(Element , [1,2,3,4] , Rest).
Element = 1,
Rest = [2, 3, 4] ;
Element = 2,
Rest = [1, 3, 4] ;
Element = 3,
Rest = [1, 2, 4] ;
Element = 4,

Rest = [1, 2, 3] ;
false.
3- schedule_helper (FirstHalf , SecondHalf , Schedule , N)

This predicate will take 3 inputs the FirstHalf which is a list of the first half of the players with the highest seed, SecondHalf which is the other half of the players, N the number of players and will return the Schedule of the corresponding round .
For ex : ?- schedule_helper([1,2],[3,4],Schedule,4).
Schedule = [game(1, 3, semi_final), game(2, 4, semi_final)] ;
Schedule = [game(1, 4, semi_final), game(2, 3, semi_final)] ;
false.
4- pick2(List , Picked , Rest , Length)
This predicate will take 2 inputs a List and Length which is by default 3 (the maximum number of game that can be played in one day) and will pick from the List all possible "smaller" lists with at maximum length of 3 and minimum length of 1 which is the Picked list and a Rest list which is the remaining of the List after Removing the Picked list .
For ex : ?- pick2([1,2,3,4],Picked , Rest , 3).
Picked = [1, 2, 3],
Rest = [4] ;
Picked = [1, 2, 4],
Rest = [3] ;
Picked = [1, 2],
Rest = [3, 4] ;
Picked = [1, 3, 4],
Rest = [2] ;
Picked = [1, 3],
Rest = [2, 4] ;
Picked = [1, 4],
Rest = [2, 3] ;
Picked = [1],
Rest = [2, 3, 4] ;
Picked = [2, 3, 4],
Rest = [1] ;
Picked = [2, 3],
Rest = [1, 4] ;
Picked = [2, 4],
Rest = [1, 3] ;
Picked = [2],
Rest = [1, 3, 4] ;
Picked = [3, 4],
Rest = [1, 2] ;
Picked = [3],
Rest = [1, 2, 4] ;
Picked = [4],
Rest = [1, 2, 3] ;
false.
5- callPick2 (Schedule , Result , Days , DaysLeft)
this predicate will take 2 inputs : a Schedule of one Round and the available Days of the whole Tournament and will return a Result which is a valid arrangement of this Schedule (Round) (with maximum 3 games per day and minimum of 1 game per day

and player 1 and 2 cannot play on the same day) and the DaysLeft after arranging the this Round.
For ex : ?- callPick2([game(1,3,semi_final) , game(2,4,semi_final)],Result , 3 , DaysLeft).
Result = [[game(1, 3, semi_final)], [game(2, 4, semi_final)]],
DaysLeft = 1 ;
Result = [[game(2, 4, semi_final)], [game(1, 3, semi_final)]],
DaysLeft = 1 ;
false.

Another solution for part 3)

ismember2(List) :

it checks if the list contains game(1,_,_)and game(2,_,_) at the same time , it returns true if the List contains the elements it returns true if not returns false .

generate (List ,X) :
it takes a List and returns a list of all possible sublists of the List starting from length 1 to 3 and the elements of (X) are sorted and they do not contain game (1,_,_) and game (2,_,_) at the same time
Example : -

generate([game(1, 5, quarter_final), game(2, 6, quarter_final), game(3, 7, quarter_final), game(4, 8, quarter_final)],X).
X = [game(1, 5, quarter_final), game(3, 7, quarter_final)] ;
X = [game(1, 5, quarter_final), game(4, 8, quarter_final)] ;
X = [game(2, 6, quarter_final), game(3, 7, quarter_final)] ;
X = [game(2, 6, quarter_final), game(4, 8, quarter_final)] ;
X = [game(3, 7, quarter_final), game(4, 8, quarter_final)] ;
X = [game(1, 5, quarter_final), game(3, 7, quarter_final), game(4, 8, quarter_final)] ;
X = [game(2, 6, quarter_final), game(3, 7, quarter_final), game(4, 8, quarter_final)] ;
X = [game(1, 5, quarter_final)] ;
X = [game(2, 6, quarter_final)] ;
X = [game(3, 7, quarter_final)] ;
X = [game(4, 8, quarter_final)].

Flat(List ,X) :-
It takes a list of lists and flatten each list in it and returns a single list of all elements in (List)

Example :-

flat([a, [b, [c, d], e]], X).
X = [a, b, c, d, e]

Intersect(List1,List2,result):-

It returns the intersection of two list in a new list and if there is no intersection it returns an empty list .

My_append(List1,List2,R):-
It makes the same function of append but when appending a list of empty list with another list it ignores the occurrence of the empty list
Example for the difference between the two appends :-

```
?- append([[]],[[a]],X).
X = [[], [a]].

?- my_append([[]],[[a]],X).
X = [[a]] ;
false.
```

Real_generate(List,Acc,ListResult):-
It takes a list and an initial Accumulator (a list of empty lists) and put the result of generate predicate on the input List in the accumulator and repeat that until the length of the accumulator equals the original length of the list then the result became the accumulator (taking in consideration that when inserting an element in the accumulator form the generate predicate it checks that this element does not found in the accumulator by flatten the accumulator and find the intersection between the accumulator and the lest generated form generate on the input list) so now we have a list of lists every list represent a day ;
Examlpe :-

```
?- real_generate([game(1, 5, quarter_final), game(2, 6, quarter_final), game(3, 7, quarter_final), game(4, 8, quarter_final)],[[]],X).
X = [[game(1, 5, quarter_final), game(3, 7, quarter_final)], [game(2, 6, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(3, 7, quarter_final)], [game(2, 6, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(3, 7, quarter_final)], [game(4, 8, quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(4, 8, quarter_final)], [game(2, 6, quarter_final), game(3, 7, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(4, 8, quarter_final)], [game(2, 6, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(4, 8, quarter_final)], [game(3, 7, quarter_final)], [game(2, 6, quarter_final)]] ;
```

```
X = [[game(2, 6, quarter_final), game(3, 7, quarter_final)], [game(1,
5, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(3, 7, quarter_final)], [game(1,
5, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(3, 7, quarter_final)], [game(4,
8, quarter_final)], [game(1, 5, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(4, 8, quarter_final)], [game(1,
5, quarter_final), game(3, 7, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(4, 8, quarter_final)], [game(1,
5, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(4, 8, quarter_final)], [game(3,
7, quarter_final)], [game(1, 5, quarter_final)]] ;
X = [[game(3, 7, quarter_final), game(4, 8, quarter_final)], [game(1,
5, quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(3, 7, quarter_final), game(4, 8, quarter_final)], [game(2,
6, quarter_final)], [game(1, 5, quarter_final)]] ;
X = [[game(1, 5, quarter_final), game(3, 7, quarter_final), game(4, 8,
quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(2, 6, quarter_final), game(3, 7, quarter_final), game(4, 8,
quarter_final)], [game(1, 5, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final), game(3,
7, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final), game(4,
8, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(3, 7, quarter_final), game(4,
8, quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final), game(3,
7, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final)], [game(3,
7, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final)], [game(3,
7, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(2, 6, quarter_final)], [game(4,
8, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(3, 7, quarter_final)], [game(2,
6, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(3, 7, quarter_final)], [game(2,
6, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(3, 7, quarter_final)], [game(4,
8, quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(4, 8, quarter_final)], [game(2,
6, quarter_final), game(3, 7, quarter_final)]] ;
```

```
X = [[game(1, 5, quarter_final)], [game(4, 8, quarter_final)], [game(2,
6, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(1, 5, quarter_final)], [game(4, 8, quarter_final)], [game(3,
7, quarter_final)], [game(2, 6, quarter_final)]] ;
X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final), game(3,
7, quarter_final)], [game(4, 8, quarter_final)]] ;
X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final), game(4,
8, quarter_final)], [game(3, 7, quarter_final)]] ;
X = [[game(2, 6, quarter_final)], [game(3, 7, quarter_final), game(4,
8, quarter_final)], [game(1, 5, quarter_final)]] ;
X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final), game(3,
7, quarter_final), game(4, 8, quarter_final)]] ;
X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final)], [game(3,
7, quarter_final), game(4, 8, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final)],
[game(3, 7, quarter_final)], [game(4, 8, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(1, 5, quarter_final)],
[game(4, 8, quarter_final)], [game(3, 7, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(3, 7, quarter_final)],
[game(1, 5, quarter_final), game(4, 8, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(3, 7, quarter_final)],
[game(1, 5, quarter_final)], [game(4, 8, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(3, 7, quarter_final)],
[game(4, 8, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(4, 8, quarter_final)],
[game(1, 5, quarter_final), game(3, 7, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(4, 8, quarter_final)],
[game(1, 5, quarter_final)], [game(3, 7, quarter_final)]] ;
;X = [[game(2, 6, quarter_final)], [game(4, 8, quarter_final)],
[game(3, 7, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(1, 5, quarter_final), game(4,
8, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(2, 6, quarter_final), game(4,
8, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(1, 5, quarter_final)],
[game(2, 6, quarter_final), game(4, 8, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(1, 5, quarter_final)],
[game(2, 6, quarter_final)], [game(4, 8, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(1, 5, quarter_final)],
[game(4, 8, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(2, 6, quarter_final)],
[game(1, 5, quarter_final), game(4, 8, quarter_final)]] ;
```

```
;X = [[game(3, 7, quarter_final)], [game(2, 6, quarter_final)],
[game(1, 5, quarter_final)], [game(4, 8, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(2, 6, quarter_final)],
[game(4, 8, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(4, 8, quarter_final)],
[game(1, 5, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(3, 7, quarter_final)], [game(4, 8, quarter_final)],
[game(2, 6, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(1, 5, quarter_final), game(3,
7, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(2, 6, quarter_final), game(3,
7, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(1, 5, quarter_final)],
[game(2, 6, quarter_final), game(3, 7, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(1, 5, quarter_final)],
[game(2, 6, quarter_final)], [game(3, 7, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(1, 5, quarter_final)],
[game(3, 7, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(2, 6, quarter_final)],
[game(1, 5, quarter_final), game(3, 7, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(2, 6, quarter_final)],
[game(1, 5, quarter_final)], [game(3, 7, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(2, 6, quarter_final)],
[game(3, 7, quarter_final)], [game(1, 5, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(3, 7, quarter_final)],
[game(1, 5, quarter_final)], [game(2, 6, quarter_final)]] ;
;X = [[game(4, 8, quarter_final)], [game(3, 7, quarter_final)],
[game(2, 6, quarter_final)], [game(1, 5, quarter_final)]] ;
false.;
```