ALGORITHM FOR CRICKET PLAYER POSITION COMBINATION

The program will read this input file and will generate a list (player_position). The list will be like [Pos_1, Pos_2, Pos_3,.....]

Each position in the list (player_position) will hold the number of possible players in that particular position. Like,

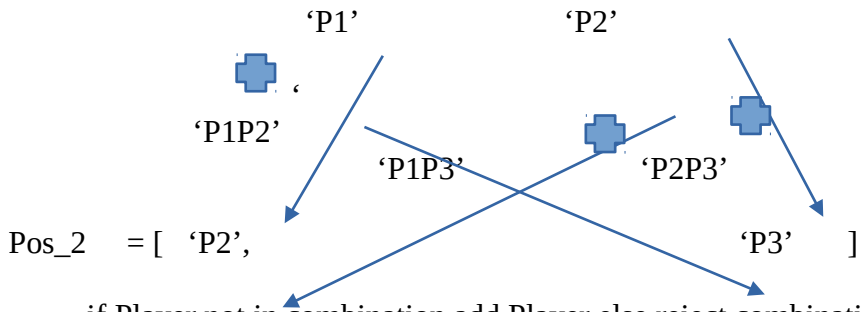
Pos_1 = [‘P1’, ‘P2’] // this means at position 1 – Player #1 and Player #2 can play
Pos_2 = [‘P2’, ‘P3’] // this means at position 2 – Player #2 and Player #3 can play
Pos_3 = [‘P1’, ‘P2’, ‘P3’] // this means at position 3– Player #1, Player #2 and Player #3 can play
.....and so on.

At Pos_1 = [‘P1’ ,‘P2’]
oldcomb = [ ]

Step 1. Generate First combination:

Insert player of position one from Pos_1 list to oldcomb list:
oldcomb = [‘P1’ ,‘P2’]

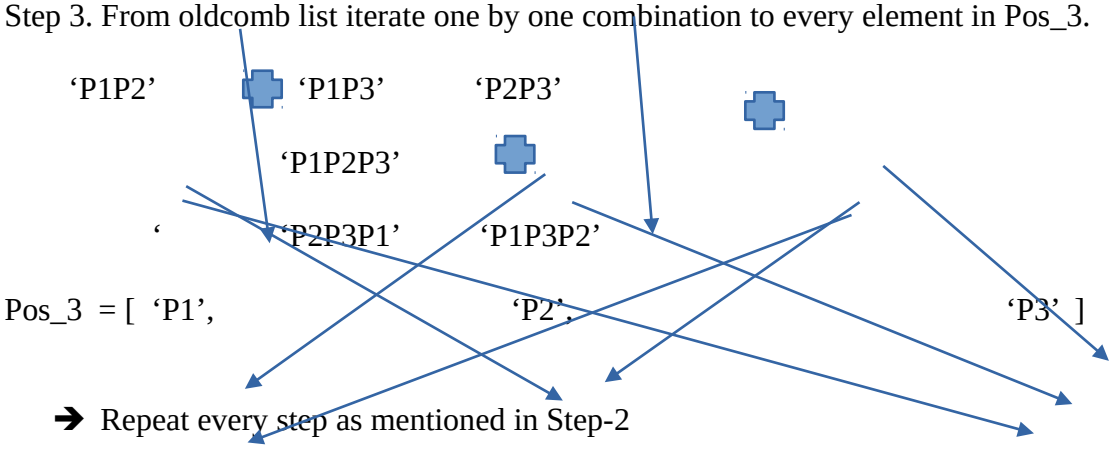
Step 2. From oldcomb list iterate one by one combination to every element in Pos_2.



➔ if Player not in combination add Player else reject combination.
➔ This symbol used to add unique player in combination

➔ This symbol is used to reject combination of same player occured before
➔ Add the new generated combination in **newcomb** list (**newcomb** = [ ])
➔ Clear the **oldcomb** list and then copy **newcomb** list to it.
➔ Repeat step-2 for all other positions
➔ So the final list as per above example will be :- oldcomb = [‘P1P2’, ‘P1P3’, ‘P2P3’]

Step 3. From oldcomb list iterate one by one combination to every element in Pos_3.



➔ Repeat every step as mentioned in Step-2

➔ So now the list would look like :- oldcomb = [‘P1P2P3’, ‘P1P3P2’ , ‘P2P3P1’
➔ So this step will continue for all other positons and will give the final list.

Step 4. At the end of the loop, copy the list **oldcomb** to **Combination** list

Step 5. Length of the **Combination** list will give us the final result (i.e.count of all possible combinations) and is finally copied to the output file outputPS7.txt

Design:-

1. readInput(inputfile) – Read file
   a. Read the data from input file passed above.
   b. Dictionary player is created
      i[0] : Key - contains player name
      [1:] : Value – contains position of that player.
2. Generate_player_position_list() --- player_position = []
   This function generates the list of player who can play from position 1 to 11.
3. Create_combination() -- This function creates combination one by one through iteration of loop by generating combination through position one list to position two list and so on till eleventh position.
   This uses dynamic programming approch :-
   a. Generate combination from first loop and store it in a list called oldcomb
   b. From previous result generate next combination and store it in temporory list called newcomb.
   c. After end of loop clear the oldcomb list and store new result from newcomb to oldcomb.
   d. At final step all combination generated in oldcomb list and store it in combination list.

   This is the idea of dynamic programming by dividing large problem into sub problem. From subproblem generate the result and store it and use result in next step. Reduce the complexity and optmize the code. This concept is used in our approach.

   i- create empty lists old comb and combination list

```
for pos-1 player in palyer_position_list-1
    add pos-1 player to oldcomb list
 for list_position in player_position_list[2-11]
      newcomb = [ ] --create empty list
      for prevcomb in oldcomb
              for newplayer in list_position
                  if newplayer not in prevcomb
                     newcombination = Add player to prevcomb --- to create further combination
                     add newcombination to newcomb list
        clear oldcomb
        copy newcomb to oldcomb
     store final result in combination list
```

TIME COMPLEXITY

Function	COMPLEXITY
readInput()	O(n)
generate_player_position_list()	O(n^2)
Create_combination()	O(n^3)
writeOutput()	O(1)
Overall program time complexity	O(n^3)