Solution QUIZ 8 (Dated: 02 Dec, 2022)
Object Oriented Programming (BSDS Spring 2022)

Roll No: ____________________ **Name:** ___________________________

Q1. Consider the given code and rewrite code using OOP concepts:

```
class Player:
 def
__init__(self,pno,pname,type,mcount,scores='',goals='',wcount1='',wcount2=''):
  self.pno = pno
  self.pname = pname
  self.type = type
  self.mcount = mcount
  self.scores = scores
  self.goals = goals
  self.wcount1 = wcount1
   self.wcount2 = wcount2

 def __str__(self):
  st = f'Player Number: {self.pno}\nPlayer Name: {self.pname}\nPlayer Type: {self.type}\nMatches: {self.mcount}'
  if self.score!='':        st += f'Scores: {self.scores}'
  if self.goals!='':        st += f'Goals: {self.goals}'
  if self.wcount1!='':      st += f'Single Matches Win Count: {self.wcount1}'
  if self.wcount2!='':      st += f'Double Matches Win Count: {self.wcount2}'
  return st
```

The parameters scores and goals are two different attributes, scores are related to Cricket or Baseball; whereas goals is a possible attribute of game Hockey or Football. Similarly, talking about str function, the if clause is clearly showing that for some player the attribute will exist and for some player the attribute will not exist. The last two, if conditions are assigning "Single Matches Win Count" and "Double Matches Win Count", again this is related to games like Badminton, Tennis, Table Tennis, Squash etc.

Both init and str functions are clearly showing that Player class has attributes of different type of players, therefore, there should be inheritance in this code:

```
class Player:
     def __init__(self,pno,pname,type,mcount):
          self.pno = pno
          self.pname = pname
          self.type = type
          self.mcount = mcount
     def  __str__(self):
          string  =  f'Player Number: {self.pno}\nPlayer Name: ‘
          string += f'{self.pname}\nPlayer'Type: {self.type}\n’
          return  string + f’Matches: {self.mcount}'

class Cricket_Player (Player):
     def __init__(self,pno,pname,type,mcount,scores='',goals='',
                                                  wcount1='',wcount2=''):
          super().__init__ ( pno,pname,type,mcount)
          self.scores = scores
     def  __str__(self):
          return super().__str__() + f'Scores: {self.scores}'
class Football_Player (Player):
     def __init__(self,pno,pname,type,mcount,scores='',goals=''
                                                  ,wcount1='',wcount2=''):
          super().__init__ ( pno,pname,type,mcount)
          self.goals = goals
  def  __str__(self):
    return super().__str__() + f'Goals: {self.goals}'

class Tennis_Player (Player):
      def __init__(self,pno,pname,type,mcount,scores='',goals='',
                                                  wcount1='',wcount2=''):
          super().__init__ ( pno,pname,type,mcount)
          self.wcount1 = wcount1
          self.wcount2 = wcount2
     def  __str__(self):
          st = super().__str__()
          st += f'Single Matches Win Count: {self.wcount1}'
          return st + f'Double Matches Win Count: {self.wcount2}'
```

Q2. Consider class Food with property [full_half, normal_spicy], SweetDish with property [full_half, cold_hot] and Drink [large_medium, cold_hot, sugar_free]. Create aggregated class Order having multiple or zero Food items, multiple or zero SweetDish items and multiple or zero Drinks. Write init and str function, assuming both exist in Food, SweetDish and Drink?

```
class Order:
      def __init__(self, count):
           self.order_list=[]
           for i in range(count):
                type = input('Enter Food Type:)
                if type == 'F':
                     self.order_list.append(Food())#assuming init has input statements
                elif type == 'S':
                     self.order_list.append(SweetDish())
                else:
                     self.order_list.append(Drink())
     def  __str__(self):
           st = ''
           for food in self.order_list:
                st += str(food) + '\n'#assuming \n missing in str of Food...
           return st
```

Q3. Rewrite question 2 with class level member dictionary with property and class name to invoke class based on property without using if-else.

```
class Order:
    order_type={'F':Food, 'S':SweetDish, 'D':Drink}
    def __init__(self, count):
        self.order_list=[]
        for i in range(count):
            object_type = input('Enter Food Type:')
            object_class = Order.order_type[object_type]
            self.order_list.append(object_class())
```