SET - A

Suppose you are given the duty to make an app for your local weather news for your city so that the local residents can get important updates about the weather on the go. But for this they need to be added as a member in the system so get the updates. Every member subscribing to the weather app must get the updates. The members can also unsubscribe themselves from the system. You need to make sure that there is only application of the weather app to ensure consistency.

1. What design pattern should be appropriate for this system?
   Answer: Singleton+Observer

```
class WeatherApp:
  _instance = None

  def __new__(cls): #2
    if cls._instance is None:
      cls._instance = super().__new__(cls)
    return cls._instance

  def __init__(self): #2
    self.members=[]

  def add_member(self,member): #2
    self.members.append(member)

  def remove_member(self,member): #2
    self.members.remove(member)

  def set_weather_update(self, update): #2
    for member in self.members:
      member.notify(update)


class Member:
  def __init__(self,name): #2
    self.name=name
    self.weatherapp = None

  def notify(self, update): #2
    print(f"{self.name} received weather update: {update}")

  def subscribe(self, weatherapp): #2
    self.weatherapp = weatherapp
    self.weatherapp.add_member(self)

  def unsubscribe(self, weatherapp): #2
    self.weatherapp = weatherapp
    self.weatherapp.remove_member(self)


w1 = WeatherApp()
w2 = WeatherApp()
print(w1)
print(w2)

member1 = Member("Alice")
member2 = Member("Bob")
member3 = Member("Samantha")


member1.subscribe(w1)
member2.subscribe(w2)
member3.subscribe(w1)

w1.set_weather_update("It might Rain Today")

member1.unsubscribe(w1)

w1.set_weather_update("It is Sunny today")
```

SET - B

Suppose you are given the duty to make an app for your local weather news for your city so that the local residents can get important updates about the weather on the go. Your app has two methods: getLocalTemp and getSuggestions. After implementing the app, you are told to merge with the central weather app of the country. The central weather app has the following methods: getCityTemp(cityname),

citySuggestions(cityname). Now you need to make necessary adjustments to the code. You also need to make sure that there is only application of the weather app to ensure consistency.

1. What design pattern should be appropriate for this system?
   Answer: Singleton+Adapter

```
class LocalWeatherApp:

  def __init__(self, central, cityname):
    self.members=[]
    self.cityName = cityname
    self.central = central

  def getLocalTemp(self):
    return self.central.getCityTemp(self.cityName)

  def getSuggestions(self):
    return self.central.citySuggestions(self.cityName)


class CentralWeatherApp:
  _instance = None

  def __new__(cls):
    if cls._instance is None:
      cls._instance = super().__new__(cls)
    return cls._instance

  def getCityTemp(self, cityname):
    return f"{cityname} Temp update: 27 Degree Celcius"

  def citySuggestions(self, cityname):
    return f"{cityname} Weather Update: It will be a sunny day!"



c1 = CentralWeatherApp()
c2 = CentralWeatherApp()

print(c1 == c2)

local = LocalWeatherApp(c1,"Jadur Shohor")

print(local.getLocalTemp())
print(local.get_suggestions())
```