NAME : MIAN AHSAN JAN
SUBJECT : MODER PROGRAMMING LANGUAGE
ID# : 13213
DATED : 23RD/SEP/2020

Q1)

Solution:

```
class Restaurant():
        def __init__(self,name,cuisine_type):
                self.name = name
                self.cuisine_type = cuisine_type
        def describe_restaurant(self):
                print(self.name ,self.cuisine_type)
        def open_restaurant(self):
                print('the',self.name, 'is opening')

restaurant = Restaurant('zct','food')
restaurant.describe_restaurant()
restaurant.open_restaurant()

#9-3
class User():
        def __init__(self,first_name,last_name,**user_info):
                self.first_name = first_name
                self.last_name = last_name
                self.user_info = user_info
        def greet_user(self):
                print('Hello,',self.first_name.title())
        def describe_user(self):
                print('first_name is ',self.first_name)
                print('last_name is ',self.last_name)
                for k,v in self.user_info.items():
                        print(k,'is',v)

user1 = User('xiaoli','li',age=12,sex='F')
user1.describe_user()
user1.greet_user()
#9-4
```

```
class Restaurant():
	def __init__(self,name,cuisine_type):
		self.name = name
		self.cuisine_type = cuisine_type
		self.number_served = 0
	def describe_restaurant(self):
		print(self.name ,self.cuisine_type,self.number_served)
	def set_number_served(self,served_num):
		self.number_served = served_num
	def increment_number_served(self,inc_served_num):
		self.number_served += inc_served_num

restaurant = Restaurant('zct','food')
restaurant.describe_restaurant()
restaurant.set_number_served(56)
restaurant.describe_restaurant()
restaurant.increment_number_served(12)
restaurant.describe_restaurant()
#9-5
class User():
	def __init__(self,first_name,last_name,**user_info):
		self.first_name = first_name
		self.last_name = last_name
		self.login_attempts = 0
		self.user_info = user_info
	def greet_user(self):
		print('Hello,',self.first_name.title())
	def describe_user(self):
		print('first_name is ',self.first_name)
		print('last_name is ',self.last_name)
		print('login_attempts is ',self.login_attempts)
		for k,v in self.user_info.items():
			print(k,'is',v)
	def increment_login_attempts(self):
		self.login_attempts+=1
	def	reset_login_attempts(self):
		self.login_attempts = 0

user1 = User('xiaoli','li',age=12,sex='F')
user1.describe_user()
user1.increment_login_attempts()
user1.increment_login_attempts()
user1.describe_user()
user1.reset_login_attempts()
user1.describe_user()
#9-6
class IceCreamStand(Restaurant):
```

```
        def __init__(self,name,cuisine_type,flavors):
                super().__init__(name,cuisine_type)
                self.flavors = flavors
        def show_flavors(self):
                for flavor in    self.flavors:
                        print(flavor)
iceCream=IceCreamStand('aa','bb',['aaa','bbb','ccc'])
iceCream.show_flavors()
class User():
        def __init__(self,first_name,last_name,**user_info):
                self.first_name = first_name
                self.last_name = last_name
                self.user_info = user_info
        def greet_user(self):
                print('Hello,',self.first_name.title())
        def describe_user(self):
                print('first_name is ',self.first_name)
                print('last_name is ',self.last_name)
                for k,v in self.user_info.items():
                        print(k,'is',v)
#9-7
class Admin(User):
        def __init__(self,first_name,last_name):
                super().__init__(first_name,last_name)
                self.privileges = ['can add post','can delete post','can ban user']

        def show_privileges(self):
                for privilege in self.privileges:
                        print(self.first_name,self.last_name,privilege)

admin = Admin('adminf','adminl')
admin.show_privileges()

#9-8
class Privileges():
        def __init__(self):
                self.privileges = ['can add post','can delete post','can ban user']


class Admin(User):
        def __init__(self,first_name,last_name):
                super().__init__(first_name,last_name)
                self.privileges = Privileges()

        def show_privileges(self):
                for privilege in self.privileges.privileges:
                        print(self.first_name,self.last_name,privilege)
```

```
admin = Admin('adminf','adminl')
admin.show_privileges()
#9-9
class Car():
        def __init__(self, make, model, year):
                self.make = make
                self.model = model
                self.year = year
                self.odometer_reading = 0
        def get_descriptive_name(self):
                long_name = str(self.year) + ' ' + self.make + ' ' + self.model
                return long_name.title()
        def read_odometer(self):
                print("This car has " + str(self.odometer_reading) + " miles on it.")
        def update_odometer(self, mileage):
                if mileage >= self.odometer_reading:
                        self.odometer_reading = mileage
                else:
                        print("You can't roll back an odometer!")
        def increment_odometer(self, miles):
                self.odometer_reading += miles
class Battery():
        """A simple attempt to simulate an electric car battery """
        def __init__(self, battery_size=70):
                """ Initialize the battery's properties """
                self.battery_size = battery_size
        def describe_battery(self):
                """ print a message describing the battery capacity """
                print("This car has a " + str(self.battery_size) + "-kWh battery.")
        def get_range(self):
                """ print a message indicating the battery's cruising range """
                if self.battery_size == 70:
                        range = 240
                elif self.battery_size == 85:
                        range = 270
                message = "This car can go approximately " + str(range)
                message += " miles on a full charge."
                print(message)
        def upgrade_battery(self):
                """ print a message describing the battery capacity """
                if self.battery_size!=85:
                        self.battery_size=85
                print("This car upgrade to " + str(self.battery_size) + "-kWh
battery.")
class ElectricCar(Car):
        """The uniqueness of electric cars """
```

```
        def __init__(self, make, model, year):
                """ initializes the properties of the parent class, and then
initializes the electric car-specific property """
                super().__init__(make, model, year)
                self.battery = Battery()
my_tesla = ElectricCar('tesla', 'model s', 2016)
print(my_tesla.get_descriptive_name())
my_tesla.battery.upgrade_battery()
my_tesla.battery.get_range()
#9-14
from random import randint


class Die():
        def __init__(self,sides=6):
                self.sides = sides
        def roll_die(self):
                print(randint(1,self.sides))
die_six = Die(6)
for i in range(10):
        die_six.roll_die()
die_ten = Die(10)
for i in range(10):
        die_ten.roll_die()
die_20 = Die(20)
for i in range(10):
        die_20.roll_die()
```

Q2

Solution:

```
def make_album(artist, title, tracks=0):
    """Build a dictionary containing information about an album."""
    album_dict = {
        'artist': artist.title(),
        'title': title.title(),
        }
    if tracks:
        album_dict['tracks'] = tracks
    return album_dict

# Prepare the prompts.
title_prompt = "\nWhat album are you thinking of? "
artist_prompt = "Who's the artist? "
```

```
# Let the user know how to quit.
print("Enter 'quit' at any time to stop.")

while True:
    title = input(title_prompt)
    if title == 'quit':
        break

    artist = input(artist_prompt)
    if artist == 'quit':
        break

    album = make_album(artist, title)
    print(album)

print("\nThanks for responding!")
```

Q3)

Solution:

A
```
car = input("What kind of car would you like? ")
print("Let me see if I can find you a " + car.title() + ".")
```

B
```
party_size = input("How many people are in your dinner party tonight? ")
party_size = int(party_size)

if party_size > 8:
    print("I'm sorry, you'll have to wait for a table.")
else:
    print("Your table is ready.")
```
C
```
number = input("Give me a number, please: ")
number = int(number)

if number % 10 == 0:
    print(str(number) + " is a multiple of 10.")
```
D
```
number = input("Give me a number, please: ")
number = int(number)

if number % 20 == 0:
    print(str(number) + " is a multiple of 20.")

else:
    print(str(number) + " is not a multiple of 20.")
```

E
```
number = input("Give me a number, please: ")
```

```
number = int(number)

if number % 30 == 0:
        print(str(number) + " is a multiple of 30.")

F number = input("Give me a number, please: ")
number = int(number)

if number % 140 == 0:
        print(str(number) + " is a multiple of 140.")
```

Q4)

Solution:

```
prompt = "\nWhat topping would you like on your pizza?"
prompt += "\nEnter 'quit' when you are finished: "

while True:
        topping = input(prompt)
        if topping != 'quit':
                print("    I'll add " + topping + " to your pizza.")
        else:
                Break
```

Output:

What topping would you like on your pizza?

Enter 'quit' when you are finished: pepperoni

I'll add pepperoni to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: sausage

I'll add sausage to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: bacon

I'll add bacon to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: quit

Q5)

Solution:

```
rivers = {

   'nile': 'egypt',
   'mississippi': 'united states',
   'fraser': 'canada',
   'kuskokwim': 'alaska',
   'yangtze': 'china',
   ‘amazon river’ :’ brazil’ ,
   ‘danube’ :  ‘germany’ ,
   ‘ganges’  :’ india’ ,
   ‘mekong ’ :’ vietnam’
   ‘zambezi’ :’ zimbabwe’
    }

for river, country in rivers.items():
    print(f"The {river.title()} flows through {country.title()}.")

print("\nThe following rivers are included in this data set:")
for river in rivers.keys():
    print(f"- {river.title()}")

print("\nThe following countries are included in this data set:")
for country in rivers.values():
    print(f"- {country.title()}")
```

Output :

The Mississippi flows through United States.
The Yangtze flows through China.
The Fraser flows through Canada.
The Nile flows through Egypt.
The Kuskokwim flows through Alaska.
The amazon river flows through Brazil.
The danube flows through Germany.
The ganges flows through India.
The mekong flows through Vietnam.
The zambezi flows through Zimbabwe.

The following rivers are included in this data set:

- ◆ - Mississippi
- ◆ - Yangtze
- ◆ - Fraser
- ◆ - Nile
- ◆ - Kuskokwim
- ◆ Amazon river
- ◆ Danube
- ◆ Ganges
- ◆ Mekong
- ◆ Zambezi

The following countries are included in this data set:

- ✓ - United States
- ✓ - China
- ✓ - Canada
- ✓ - Egypt
- ✓ - Alaska
- ✓ Brazil
- ✓ Vietnam
- ✓ Germany
- ✓ Zimbabwe
- ✓ India