Name: SRADHA KEDIA
Date and time of Examination: 9:30am - 12:30pm ; 27/03/2021
Examination Roll no: 2023475053
Name of the Programe: MCA
Semester: Ist
Unique Paper Code: 223401101
Title of the Paper - Object Oriented Programming
Email-ID- 200083@cs.du.ac.in
Mobile no.— 8840502121
No. of Pages— 7

Date
Page 01

2023475705 3

Question 5:

```
class Date:

    '''
        constructor : __init__
        attributes : __day, __mon, __year
        functions: __str__, __del__, __le__, objcount, daymonyear
        Class variable : count
    '''

    count = 0       # to count object
    def __init__(self, __day, __mon, __year):
        if __day < 0 or __day > 31:        # less than 0 or more than 31
            raise Exception("Days between 0-31 should be there")
        if __mon < 0 or __mon > 12:
            raise Exception("Months should be between 1 to 12")

    # rasing exception when input is invalid.

    Date.count += 1    # to count object, incrementing
    self.__day = __day
    self.__month = __mon
    self.__year = __year

    def __str__(self):
    # overloading the str dunder function
        return f"{self.__day}/{self.__month}/{self.__year}"

    def __del__(self):
        # to delete, overloading del
```

```
Date.count -= 1

def __le__(self, obj2):
    # overloading less than equal to (le) dunder method.

    if self.__year > obj2.year:
        return False
    if self.__month > obj2.month and self.__year == obj2.year:
        return False
    if self.__day > obj2.day and self.__month == obj2.mon:
        return False
    return True

# if the obj2 has proper format it goes in no if
# statement and returns false else return true if all
# correct.

@classmethod
def objcount(cls):          # class method
    return cls.count        # return no. of objects created.

@property                   # property decorator.
def day(self):              # getter
    return self.__day       # return day

@property                   # property decorator
def month(self):            # getter
    return self.__month     # return month
```

2023475053

@property
def year(self):
    return self.__year

property decorator
getter
returns year.

class Employee:

"""
class variable : count
constructor : __init__
attributes = name, hiredate, salary
functions : __str__, __del__, objcount, name, hiredate,
hiredateobj, salary, experience
"""

count = 0
def __init__(self, name, hiredate, salary):
    Employee.count += 1 # incrementing count when object
    self.__name = name
    self.__hiredate = hiredate
    self.__salary = salary

def __str__(self):
    return f""" Employee details are :
        Name — {self.__name}
        hiring date — {self.__hiredate}
        Salary — {self.__salary}
    """

2023475053

```
def __del__(self):

    # overloading __del__
    Employee.count -= 1

    # decrementing count when object is removed.

@classmethod                          # class method
def objcount(cls):
    return cls.count

@property                             # property decorator
def name(self):
    return self.__name

@property                             # property decorator
def hiredateobj(self):                # getter
    return self.__hiredate

@property
def salary(self):                     # getter
    return self.__salary

@salary.setter                        # setter
def salary(self, newsal):
    self.__salary = newsal

def experience(_empobj, _dateobj):
    hiring_date = _empobj.hiredateObj
```

2023475705 3

```
if _dateobj <= hiring_date:        # checking condition
    raise Exception ("The hiring date should be less than
                      current date")

if hiring_date.day <= _dateObj.day:        # comparing day
    date = _dateObj.day - hiring_date.day
    month = _dateObj.month - hiring_date.month

else:                                  # if day is correct then
    day = _dateobj.day - hiring_date.day
    month = _dateobj.month - hiring_date.month

    year = _dateObj.year - hiring_date.year

    return Date( day, month, year)

employees = []                         # empty list

# menu driven

print (""" \n Enter 1 to create new employee object
          Enter 2 to print all the employee with experience
          Enter 3 to get total employees
          Enter 4 to get total date objects
          Enter 0 to terminate
      """)

choice = int (input ("Enter your choice :").strip()) # input choice
while ( choice != 0):
```

2023475705 3

```
if choice == 1:
    name = input("Enter the name to be pushed:")
    print("Enter the hiring date:")
    day = int(input("Enter hiring day:"))
    month = int(input("Enter hiring month:"))
    year = int("input("Enter hiring year:"))
    salary = int(f"input(" Enter salary of Employee {name}:"))

    try:
        employees.append(Employe(name, Date(day, month, year),
                                 salary))

    except Exception as e:
        print(e)

elif choice == 2:
    for i in employees:
        print(i, "\n")
elif choice == 3:
    print("Total no. of employee objects:", Employe.objcount())
elif choice == 4:
    print("Total no. of date objects:", Date.objcount())

elif choice == 0:
    break                    # to terminate break
else:
    print("Enter the invalid choice:")

choice = int(input("Enter choice"))
```