Department of Computer Science & Engineering

University of Asia Pacific (UAP)

Program: B.Sc. in Computer Science and Engineering

Final Examination **Fall 2020** **3rd Year 1st Semester**

Course Title: Object Oriented Programming II: Visual and Web Programing

Course Code: CSE 309 **Credits: 3**

Full Marks: 120* (Written) **Duration: 2 Hours**

* Total Marks of Final Examination: 150 (Written: 120 + Viva: 30)

Instructions:

1. There are **Four (4)** Questions. Answer all of them. All questions are of equal value. Part marks are shown in the margins.
2. Non-programmable calculators are allowed.

1. a) You and your friends love gardening. So you plan to design a web application where people can upload their plants' photos with description and they can also buy or sell plants from the website. When a user is uploading his plants' details, he can choose either the plant is for sale or not. He can also set the price. The necessary model is already created as follows to store the details of the plants. **[2 + 6 + 6 + 6 = 20]**

models.py

```python
from django.db import models
from django.contrib.auth.models import User

# Create your models here.


class Plant(models.Model):
    photo = models.ImageField(upload_to='photos/images/',
                              blank=False,null=False)
    plant_name = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    date_of_upload = models.DateField(auto_now=True)
    is_for_sale = models.BooleanField(default=False)
    price = models.IntegerField(default=None,null=True)

    user = models.ForeignKey(User,on_delete=models.CASCADE)


```

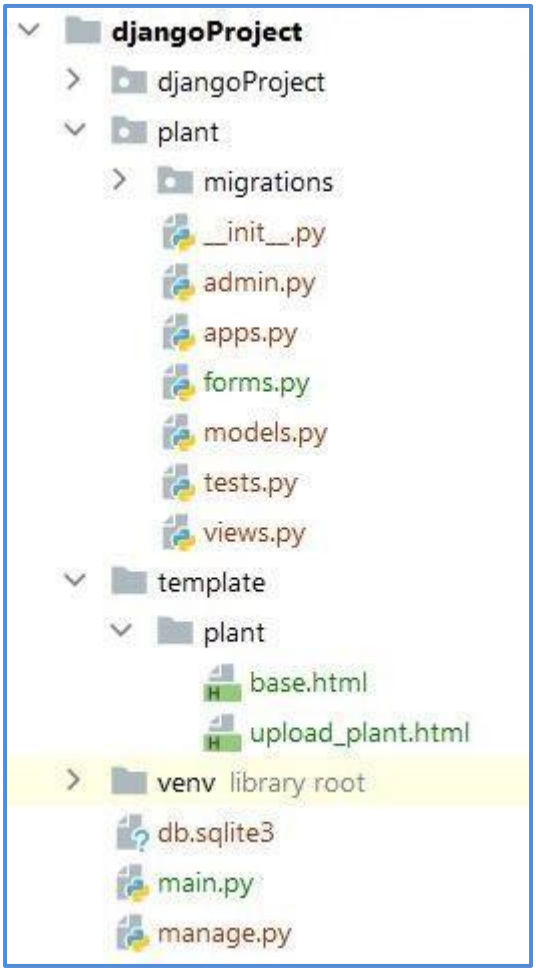
Now you have to implement the necessary urls, forms, views and templates so that a user can upload the mentioned details of his plant (**using Django forms**)

Now mention your changes to the following files-

i. urls.py
ii. plant/forms.py
iii. plant/views.py
iv. template/plant/upload_plant.html (assuming there is a base.html already implemented)

Add all the necessary imports to your files.

For your convenience, the project structure is given below.



b) After each successful upload of a plant's details, you want to send an email to the user telling that the data has been uploaded successfully. **[7+3 = 10]**

Now mention the code you need to write with necessary imports and the location where you will add this code. For your convenience, put line no in the file from **Q1.a** where you will add

this code. For example you can write, "the code will be added in between line no X and line no Y in A.py file." If you have not answered Q1.a then you do not have to mention the location of the code. You will get partial marks.

Assume that, ***User*** model has the email of a user as the ***"email"*** field. The necessary settings have already been added in the **settings.py** file as follows-

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = os.environ.get('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_HOST_PASSWORD')
```

NB: You do not need to write the whole code again. Only the code to send the email mentioning the location of the code.

2. a) As you are developing a site where users can upload, buy and sell their plants (as question no 1), now you need to add a feature where other users can rate a particular plant of a user and add comment(s) below it. **[10 +10 =20]**

Assume that the ***Plant*** model contains the information of **plants** and the ***User*** model contains the information of **users**.

To implement this feature, you need to write the following two models. Add appropriate keys while implementing the models.

i. ***Rating*** – this model will contain the ratings given to a plant and the user who gave it. This model will have many to one relationship with ***Plant*** model and many to one relationship with ***User*** model. (each row will store individual rating, not average ratting)

ii. ***Comment*** - this model will contain the comments given to a plant and the user who gave it. This model will have many to one relationship with ***Plant*** model and many to one relationship with ***User*** model.

b) Based on the models you wrote in **Q2.a** and the ***Plant*** model from **Q1.a** write necessary codes to extract the following data from the database- **[5 + 5 = 10]**

i. Get all the comments given to a plant having id = ***X***

ii. Get all the plants that have ***"cactus"*** in their name (case insensitive) and have a price lower than ***Y*** taka.

Here,
X = last two digits of your role +1
Y= X * 11
If your roll is ***18201002*** then, ***X = 3 , Y = 33***

3. a) Suppose you and your younger brother went to an animal store to get two **pets**. There were different sections of **cats** and **dogs**. From there, you got a cat and named it **"Garfield"**. Your brother got a dog and named it **"Odie"**. **[5 x 4 = 20]**

Now according to the concept of object oriented programming, categorize the terms – ***Pet, Cat, Dog, Garfield*** and ***Odie*** into **abstract class**, **class** and **object** with a **brief explanation**.

b) Describe the concept of method overloading and method overriding in object oriented programming (OOP) and how it is handled in python OOP with short examples. **[5 + 5 = 10]**

4. a) In universities, a teacher teaches the class and the students take lessons from teachers. But if you are a teaching assistant, you will have to act as a teacher while assisting your professor and also act as a student while working under your professor. Furthermore, both the teachers and students are persons as well. **[4 x 5 = 20]**

Now write the following classes giving them accurate attributes and inheritance relations. No need to add the getter setter methods of the attributes. Only show the constructors of each class.

i. Person
ii. Teacher
iii. Student
iv. TeachingAssistant

Mention the types of inheritance relations between the classes.

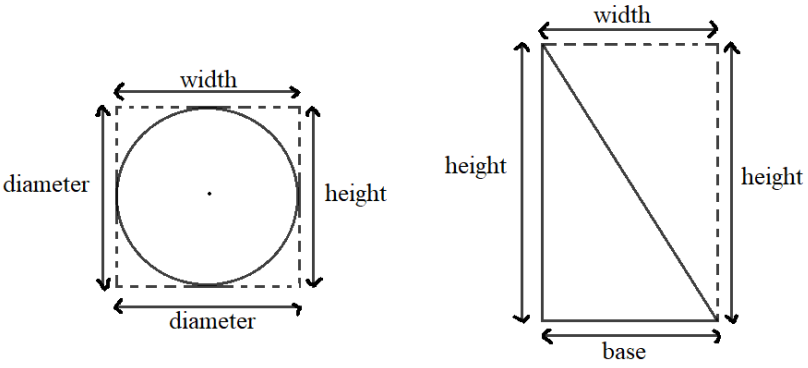
or,

We have an abstract class **Shape** which has two private variables – height and width, has two abstract methods **getArea()** and **getPerimeter()** and two getter methods – **getHeight()** and **getWidth().** **[4 x 5 = 20]**

We have three more non-abstract classes – **Circle**, **Triangle** and **Rectangle** which inherit the **Shape** class. These classes do not have any additional attributes. Assume that, the triangle will always be a right-angled triangle.

The constructors will have the parameters as follows –
Shape – height, width
Circle – diameter
Rectangle – height, width
Triangle – height, base

Now implement all four of the above classes.



NB: the dotted lines (- - -) represent the shape

b) Write a function that will take a list of integer numbers as a parameter and return the number from the list which has the maximum square value. Here the list may contain both positive and negative numbers. **[10]**

or,

Write a function that will take two lists as parameters. The first list will contain some persons' names and the second list will contain the ages of those persons in order. The function will return a dictionary containing only the persons who are 18 years old or more. Here the persons' names will be the keys of the dictionary and ages will be the values of those keys. **[10]**