**SULIT**



BAHAGIAN PEPERIKSAAN DAN PENILAIAN

JABATAN PENDIDIKAN POLITEKNIK DAN KOLEJ KOMUNITI

KEMENTERIAN PENDIDIKAN TINGGI

JABATAN TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

PEPERIKSAAN AKHIR

**SESI II : 2023/2024**

**DFP40203: PYTHON PROGRAMMING**

|  |  |  |  |
| --- | --- | --- | --- |
| Jenis item | Politeknik Pengubal | Bil. Perlu Gubal | Bilangan Set |
| Subjektif : 3 Soalan | PSP | 1 Set | Set 1 |

**JANGAN BUKA KERTAS SOALAN INI SEHINGGA DIBERITAHU**

**OLEH KETUA PENGAWAS**

**SULIT**

**INSTRUCTION:**

This section consists of **THREE (3)** practical questions. Answer ALL questions.

**QUESTION:** Complete the program below based on this question.

The ABC Compound Unit is the unit responsible for handling the review of parking summonses in Bandar Perda. As a programmer, help the unit develop a summons payment calculation application so that users can more easily check the payment rates of several summonses charged to them and the total amount of all summonses that need to be paid. Each summon can get a 20% discount if the unit gives a discount on the summon. Table 1 is used as a guide for the payment of the summons according to the number of days from the date of the summons being issued. The application shall collect user information, such as name, car number, date of summons, and summons fee charged as proof of review and payment.

**Formula for summon discount 20%: charge = charge – (charge \* 0.20)**

Table 1: Summons charge rate

|  |  |
| --- | --- |
| **Category** | **Summons Charge (RM)** |
| Payment in 14 days | 15.00 |
| Payment after 14 days | 20.00 |
| Payment after 28 days | 25.00 |
| Payment after 60 days | Zero charge and Blacklist name |

|  |
| --- |
| import time  from datetime import date  class ABCSummonsUnit:  def \_\_init\_\_(self,totalSummons):  self.totalSummons = totalSummons  def coundDay(self, summonsDate, today):  date\_format = "%Y-%m-%d"  a = time.mktime(time.strptime(summonsDate, date\_format))  b = time.mktime(time.strptime(today, date\_format))  delta = b - a    return int(delta / 86400)  **#def calculatePrice(self, discount, duration):**  **# return charge**  def summonCalculate(self):  print("\nSummons "+str(i+1)+": ")  today = str(date.today())  print("Today's date: ", today)  summonsDate = input("Date of issue of summons (YYYY-MM-DD): ")  duration = self.coundDay(summonsDate, today)  print("Duration of summonses in days",duration)  discount = input("Does ABC Summons Unit offer a discount for this summons?: ")  payment = self.calculatePrice(discount, duration)  return payment  name = input("Name: " )  vehicleNumber = input("Vehicle number: " )  totalSummons = int(input("Number of summons to be paid: "))  totalPayment = 0  for i in range (totalSummons):  summonsPrice = user.summonCalculate()  totalPayment = totalPayment + summonsPrice    print("\nTotal Summons Amount (RM): ", totalPayment) |

Figure 1: *main.py*



Figure 2: Example output

CLO1

P2

1. Read the question below to guide you to complete the program.
2. Explain the error that be found in code at *main.py* file and solve it. **(3 Marks)**
3. Complete the function calculatePrice code at *main.py* program to meet the summons charge rate that have been set in Table 1 also can produce output like Figure 2. **(10 Marks)**

CLO1

P4

1. Constructs two module files named as *Day.py* and *Price.py* for the functions coundDay and calculatePrice to make the code *main.py* simpler and can also produce output like Figure 2.  **(5 Marks)**

CLO1

P4

1. Construct code in the *main.py* file to demonstrate that the matrix number folder contains the files *main.py*, *Price.py* and *Day.py*. **(4 Marks)**

CLO1

P4

1. The last step is to put in place a feature that stores all data in a database using whichever SQL system you choose. You must write a new application that is intended to collect and store the necessary data to achieve this. **(10 marks)**

CLO1

P4

1. Builds a database named ABCSummonsUnit.

CLO1

P4

1. Make a table summonInfo with given attributes as in table below with vehicle number as the primary key.

Table 2: Attributes table Summons Info

|  |  |
| --- | --- |
| **Attribute** | **Data type** |
| name | varchar |
| vehicle number | varchar |
| summons date | varchar |
| duration | int |
| discount | varchar |
| amount | varchar |

1. The summons application is almost finished being developed. To make it easier for users to check the payment of the summons, you need to develop a Graphical User Interface (GUI) for this application. This application allows it to calculate the summons due for one summon at a time. Utilize your knowledge and proficiency with programming to complete this application. **(18 marks)**

|  |
| --- |
| from tkinter import \*  from tkinter import ttk  import Price as \_\_\_\_\_X\_\_\_\_\_  import Day as \_\_\_\_\_X\_\_\_\_\_  from datetime import date  #import connection as data  #db = \_\_\_\_\_X\_\_\_\_\_.mydb.cursor()  def message():  root = Tk()  root.geometry("250x100")  w = Label(root, text ='ABC Summons Unit', font = "20")  w.pack()  message = Message(root, text = "\*\*Success\*\* \n")  \_\_\_\_\_X\_\_\_\_\_.pack()  root.mainloop()  def summonCalculate():  today = str(date.today())  summonDate = summonDateR.get()  duration = d.coundDay(summonDate, \_\_\_\_\_X\_\_\_\_\_ )  durationL.config(text=duration)  discount = discountR.get()  payment = p.calculatePrice(\_\_\_\_\_X\_\_\_\_\_, duration)  ammountL.config(text=payment)  def insert():  name = nameE.get()  vehicleNumber = vehicleNumberE.get()  summonDate = summonDateR.get()  duration = durationL.cget("text")  discount = discountR.get()  ammount = ammountL.cget("text")    try:  db. \_\_\_\_\_X\_\_\_\_\_ ('INSERT INTO summonInfo (name, \_\_\_\_\_X\_\_\_\_\_ , summonDate, duration, discount, ammount) VALUES (%s,%s,%s,%s,%s,%s)',  (name, vehicleNumber, summonDate, duration, discount, ammount))  data.mydb.commit()  message()    except:  data.mydb.rollback()  **#def delete():**  **# vehicleNumber = vehicleNumberE.get()**    window = Tk()  window.title("ABC Summons Unit")  up=Label(window, text="------ Summons Form ------", font=("Arial Bold", 10))  up.grid(columnspan=3, row=3, sticky='S')  namelbl=Label(window, text="Name : ")  namelbl.grid(column=0, row=4, sticky='W')  nameE=Entry(window, width=30)  nameE.grid(column=1, columnspan=2, row=4, sticky='W')  vehicleNumberlbl=Label(window, text="Vehicle Number : ")  vehicleNumberlbl.grid(column=0, row=5, sticky='W')  vehicleNumberE=Entry(window, width=30)  vehicleNumberE.grid(column=1, columnspan=2, row=5, sticky='W')  middle=Label(window, text="--"\*20)  middle.grid(columnspan=3, row=8, sticky='S')  calculatelbl=Label(window, text="Summons Calculate:", font=("Arial Bold", 10))  calculatelbl.grid(column=0, row=9, sticky='W')  todayR = date.today()  todaylbl=Label(window, text="Date of today: ")  todaylbl.grid(column=0, row=10, sticky='W')  todayE=Label(window, text=f"{todayR:%Y-%m-%d}", textvariable=todayR, width=13, borderwidth=2, relief="groove")  todayE.grid(column=1, row=10, sticky='W')  summonDateR = StringVar()  summonDatelbl=Label(window, text="Summons Date: (YYYY-MM-DD)")  summonDatelbl.grid(column=0, row=11, sticky='W')  summonDateE=Entry(window, width=15, textvariable=summonDateR)  summonDateE.grid(column=1, row=11, sticky='W')  discountR = StringVar()  discountlbl=Label(window, text="Discount for this summons?:")  discountlbl.grid(column=0, row=12, sticky='W')  discountC = ttk.Combobox(window, textvariable=\_\_\_\_\_X\_\_\_\_\_, values=["YES", "NO"], width=6)  discountC.grid(column=1, row=12, sticky='W')  durationlbl=Label(window, text="Duration of summonses in days:")  durationlbl.grid(column=0, row=13, sticky='W')  durationL=Label(window, width=8, borderwidth=2, relief="groove")  durationL.grid(column=1, row=13, sticky='W')  down=Label(window, text="--"\*20)  down.grid(columnspan=3,row=14, sticky='S')  ammountlbl=Label(window, text="Total Summons Amount (RM):")  ammountlbl.grid(column=0, row=15, sticky='W')  ammountL=Label(window, text='', width=8, borderwidth=2, relief="groove")  ammountL.grid(\_\_\_\_\_\_\_\_\_X\_\_\_\_\_\_\_\_\_\_)  last=Label(window, text="--"\*20)  last.grid(columnspan=3,row=16, sticky='S')  **#menubar = Menu(window)**  **#filemenu = Menu(menubar, tearoff=0)**  **#window.config(menu=menubar)**  window.geometry("370x300")  window.mainloop() |

Figure 2: *ABC\_GUI\_Summon.py*

CLO1

P4

1. Manipulates **X** with suitable answer to complete the program above to get a GUI version of the program. **(10 Marks)**

CLO1

P4

1. Perform the ammount text color to red color.

CLO1

P4

1. Construct a menu file like figure 3 for calculate, submit, and delete menu.

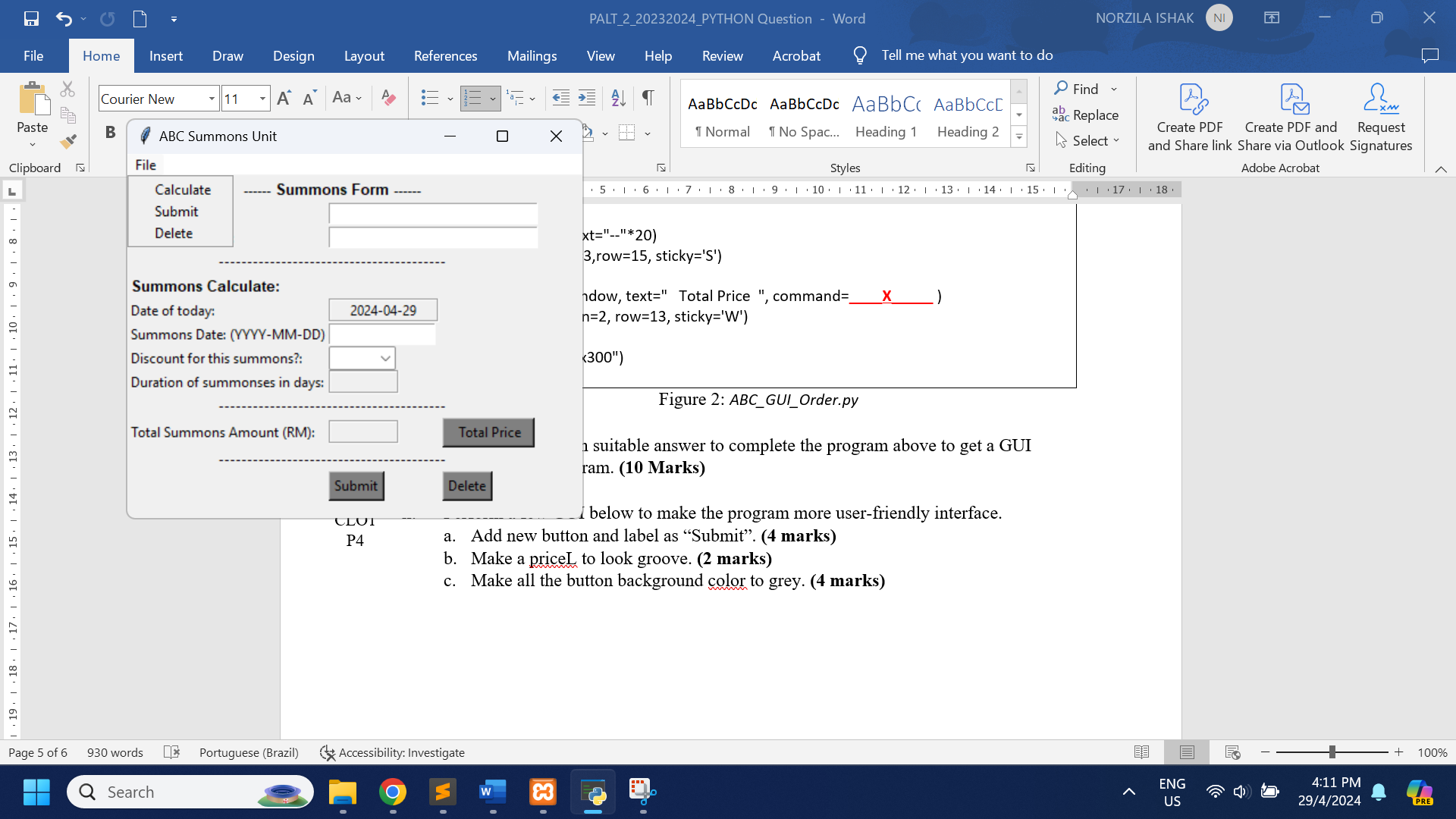


Figure 3: menu file

CLO1

P4

1. To make this application function better, perform a menu file by connecting the menus with the relevant functions.
2. Construct the full code for the delete function, including handling exceptions and a message function to let the user know that the data has been successfully deleted.

CLO1

P4