**EXAMPLE 1:**

The vaccine registration system is a system that will check the age and eligibility status of Malaysian citizens to receive the vaccine based on the eligibility schedule provided at Table 1.

|  |  |  |
| --- | --- | --- |
| Age | Status | Remarks for Vaccine |
| >60 | Senior Citizen | Eligible |
| 30-59 | Adult | Eligible |
| 18-29 | Teenagers | Eligible |
| 1-17 | Children | Not eligible |
| Not Citizen | | Not eligible |

Table 1: Vaccine eligibility

1. Refer to the code at Figure 1. Save code as example1.py, complete the code and display the output.

|  |
| --- |
| 1. from tkinter import\* 2. from tkinter import messagebox 3. import example1\_DB as data 4. import example1View as viewdata 5. db = data.connectionDB() 6. window = Tk() 7. window.title("VACCINE REGISTRATION") 8. def radio\_sel(): 9. citizen = citizenR.get() 10. age = ageR.get() 11. if citizen == "Yes": 12. if \_\_\_\_\_\_\_\_\_\_\_\_\_\_: 13. ageStatus = "Senior Citizen" 14. remarks = "Eligible for Vaccine" 15. elif \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_: 16. ageStatus = "Adult" 17. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 18. elif age>=18 and age<30: 19. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21. elif \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_: 22. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 23. remarks = "Not eligible for Vaccine" 24. else: 25. ageStatus = "None" 26. remarks = "Invalid Age" 28. categoryE.configure(text=""+ageStatus) 29. vacStatusE.configure(text=""+remarks) 30. else: 31. categoryE.configure(text="None") 32. vacStatusE.configure(text="You are not Malaysian citizen") 33. def insert(): 34. name = nameE.get() 35. icnumber = icnumberE.get() 36. age = ageR.get() 37. citizen = citizenR.get() 38. Diabetes = DiabetesC.get() 39. Hyper = HyperC.get() 40. Heart = HeartC.get() 41. ageStatus = categoryE.cget("text") 42. vacStatus = vacStatusE.cget("text") 43. mesej = messagebox.askquestion("Submit","Are you sure to Submit?") 44. if mesej == 'yes': 45. try: 46. db.cur.execute('INSERT INTO \_\_\_\_\_\_\_\_\_\_\_\_ (icnumber, \_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_, citizen, \_\_\_\_\_\_\_\_\_\_\_\_\_, Hyper, \_\_\_\_\_\_\_\_\_\_\_\_, ageStatus, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) VALUES (%s,\_\_\_,%s,\_\_\_,\_\_\_\_,%s,\_\_\_\_,\_\_\_\_\_\_,%s)', 47. (\_\_\_\_\_\_\_\_, name, age, \_\_\_\_\_\_\_\_, Diabetes, \_\_\_\_\_\_\_, Heart, \_\_\_\_\_\_\_\_\_, vacStatus)) 48. \_\_\_\_\_\_\_\_\_\_.conn.commit() 49. messagebox.showinfo("Record Success","Record inserted") 50. # print(db.cur.rowcount, "record inserted!") 51. except: 52. db.conn.rollback() 53. else: 54. messagebox.showerror("Record","Record not inserted") 55. def selectAll(): 56. if(allC.get()==1): 57. c1.select() 58. c2.select() 59. c3.select() 60. else: 61. c1.deselect() 62. c2.deselect() 63. c3.deselect() 64. #Data citizen 65. citizenlbl=Label(window, text="Malaysian citizen : ", font=("Arial Bold",11)) 66. citizenlbl.grid(column=0, row=1, sticky='W') 67. #radio button 68. citizenR = StringVar() 69. yesR = Radiobutton(window, text="Yes", variable=citizenR, value="Yes", command=radio\_sel) 70. yesR.grid(column=1, row=1, sticky='W') 71. noR = Radiobutton(window, text="No", variable=citizenR, value="No", command=radio\_sel) 72. noR.grid(column=1, row=2, sticky='W') 73. lbl1=Label(window, text="----- Please complete the information below -----", font=("Arial Bold", 12)) 74. lbl1.grid(columnspan=3, row=3, sticky='W') 75. #Data name 76. namelbl=Label(window, text="Name : ", font=("Arial Bold",11)) 77. namelbl.grid(column=0, row=4, sticky='W') 78. nameE=Entry(window, width=30) 79. nameE.grid(column=1, row=4, sticky='W') 80. #Data icnumber 81. icnumberlbl=Label(window, text="IC Number : ", font=("Arial Bold",11)) 82. icnumberlbl.grid(column=0, row=5, sticky='W') 83. icnumberE=Entry(window, width=30) 84. icnumberE.grid(column=1, row=5, sticky='W') 85. #Data age 86. ageR = IntVar() 87. agelbl=Label(window, text="Age : ", font=("Arial Bold",11)) 88. agelbl.grid(column=0, row=6, sticky='W') 89. ageE=Entry(window, width=30, textvariable=ageR) 90. ageE.grid(column=1, row=6, sticky='W') 91. lbl2=Label(window, text="Do you have the following diseases :", font=("Arial Bold", 11)) 92. lbl2.grid(columnspan=3, row=8, sticky='W') 93. #Data diseases 94. #checkbox 95. DiabetesC=StringVar(window) 96. c1=Checkbutton(window,text='Diabetes Mellitus',onvalue="Yes", offvalue="No",variable=DiabetesC) 97. c1.grid(column=0, row=9, sticky='W') 98. HyperC=StringVar(window) 99. c2=Checkbutton(window,text='Hypertension',onvalue="Yes", offvalue="No",variable=HyperC) 100. c2.grid(column=0, row=10, sticky='W') 101. HeartC=StringVar(window) 102. c3=Checkbutton(window,text='Heart Disease',onvalue="Yes", offvalue="No",variable=HeartC) 103. c3.grid(column=0, row=11, sticky='W') 104. allC=IntVar(window) 105. c4=Checkbutton(window,text='Select All',variable=allC, command=selectAll) 106. c4.grid(column=0, row=12, sticky='W') 107. #Button to check status 108. btnCheck=Button(window, text="Check Status", fg="black", font=("Arial Bold",10), command=radio\_sel) 109. btnCheck.grid(column=1, row=14) 110. lbl3=Label(window, text="Your Status :", font=("Arial Bold", 12)) 111. lbl3.grid(column=0, row=16, sticky='W') 112. #Data category 113. categorylbl=Label(window, text="You are : ", font=("Arial Bold",11)) 114. categorylbl.grid(column=0, row=17, sticky='W') 115. categoryE=Label(window, text='', width=25, font=("Arial Bold", 10), borderwidth=2, relief="groove") 116. categoryE.grid(column=1, row=17, sticky='W') 117. #Data vaccine status 118. vacStatuslbl=Label(window, text="Vaccine status : ", font=("Arial Bold",11)) 119. vacStatuslbl.grid(column=0, row=18, sticky='W') 120. vacStatusE=Label(window, text='', width=25, font=("Arial Bold", 10), borderwidth=2, relief="groove") 121. vacStatusE.grid(column=1, row=18, sticky='W') 122. #Button to submit data 123. btnInsert=Button(window, text="Submit", fg="black", font=("Arial Bold",10), command=insert) 124. btnInsert.grid(column=1, row=20) 125. #Button to new window to view list data 126. btnView=Button(window, text="View", fg="black", font=("Arial Bold",10), command=viewdata.open) 127. btnView.grid(column=1, row=22) 128. window.geometry("400x450") 129. window.mainloop() |

Figure 1: example1.py

1. Complete code at Figure 2 to create a database name as vacciendb and table listcitizen(icnumber, name, age, citizen, Diabetes, Hyper, Heart, ageStatus, vacStatus). Save code as example1\_DB.py, run code and display the output. [CLO1, P4]

|  |
| --- |
| 1. import mysql.connector 2. mydb = mysql.connector.connect( 3. host="localhost", 4. user="root", 5. password="" 6. ) 7. mycursor = mydb.cursor() 8. #create database 9. mycursor.execute("**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**") 10. dataBase = mysql.connector.connect( 11. host = "localhost", 12. user = "root", 13. password = "", 14. database = "**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**" 15. ) 16. #create table 17. cursorObj = dataBase.cursor() 18. vaccine = """CREATE TABLE listCitizen ( 19. **\_\_\_\_\_\_\_\_\_\_** VARCHAR(20) NOT NULL, 20. name **\_\_\_\_\_\_\_\_\_\_\_\_\_** , 21. **\_\_\_\_\_\_\_\_\_\_** VARCHAR(3), 22. **\_\_\_\_\_\_\_\_\_\_** VARCHAR(3), 23. **\_\_\_\_\_\_\_\_\_\_** VARCHAR(30), 24. Hyper VARCHAR(30), 25. Heart VARCHAR(30), 26. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** , 27. vacStatus VARCHAR(50), 28. PRIMARY KEY (**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**) 29. )""" 30. cursorObj.execute(vaccine) 31. dataBase.close() |

Figure 2: example1\_DB.py

1. Manipulate code example1\_DB.py at Figure 2, create one class name as connVacDB to make it as database connection.

|  |
| --- |
| 1. import mysql.connector 2. class connVacDB: 3. def \_\_init\_\_(self): 4. print("...Successful connection to database...") 5. self.connDB = mysql.connector.connect(host="localhost", user="root", password="") 6. self.conn = mysql.connector.connect(host="localhost", user="root", password="", database="vacciendb") 7. self.cur = self.conn.cursor() 8. self.conn.commit() |

1. Refer to the code at Figure 3. Save code as example1View.py, run code and display the output.

|  |
| --- |
| 1. from tkinter import \* 2. from tkinter import ttk 3. import tkinter as tk 4. import LabExe4\_DB as data 5. db = data.connectionDB() 6. def open(): 7. db.cur.execute("SELECT \* FROM listcitizen") 8. result = db.cur.fetchall() 9. my\_w=Tk() 10. my\_w.geometry('900x400') 11. my\_w.title("View All Data") 13. trv=ttk.Treeview(my\_w,selectmode='browse') 14. trv.grid(row=1,column=1,padx=20,pady=20) 15. trv["columns"]=("1","2","3","4","5","6","7","8","9") 16. trv['show']='headings' 17. trv.column("1",width=80,anchor='c') 18. trv.column("2",width=80,anchor='c') 19. trv.column("3",width=80,anchor='c') 20. trv.column("4",width=80,anchor='c') 21. trv.column("5",width=100,anchor='c') 22. trv.column("6",width=100,anchor='c') 23. trv.column("7",width=100,anchor='c') 24. trv.column("8",width=80,anchor='c') 25. trv.column("9",width=150,anchor='c') 26. trv.heading("1",text="Ic Number") 27. trv.heading("2",text="Name") 28. trv.heading("3",text="Age") 29. trv.heading("4",text="Citizen") 30. trv.heading("5",text="Diabetes Mellitus") 31. trv.heading("6",text="Hypertension") 32. trv.heading("7",text="Heart Disease") 33. trv.heading("8",text="Age Status") 34. trv.heading("9",text="Vaccien Status") 35. for dt in result: 36. print(dt) 37. trv.insert("",'end',id=dt[0],values=(dt[0],dt[1],dt[2],dt[3],dt[4],dt[5],dt[6],dt[7],dt[8])) 38. my\_w.mainloop() |

Figure 3: example1View.py

**EXAMPLE 2:**

Referring to the loan eligibility requirements in table 1. As a python programmer, you need to help ABC bank to develop a Python program to check and calculate the maximum loan eligibility rate based on salary borrower. The bank only allows each borrower to make a maximum loan of RM100,000.00 (one hundred thousand). Refer to Figure1 and Figure 2 as output process. Use the given formula for calculation process.

**Formula:** Loan Eligibility = Loan Amount - (Loan Amount - (Loan Amount \*Percent))

|  |  |
| --- | --- |
| **Salary** | **Loan Eligibility Rate Percentage** |
| More RM8000.00 | 100% = 1 |
| Between RM5000.00 and RM 8000.00 | 70% = 0.7 |
| Between RM3000.00 and RM 5000.00 | 50% = 0.5 |
| Less than RM3000.00 | 30% = 0.3 |

Table 1: ABC Bank loan eligibility requirements

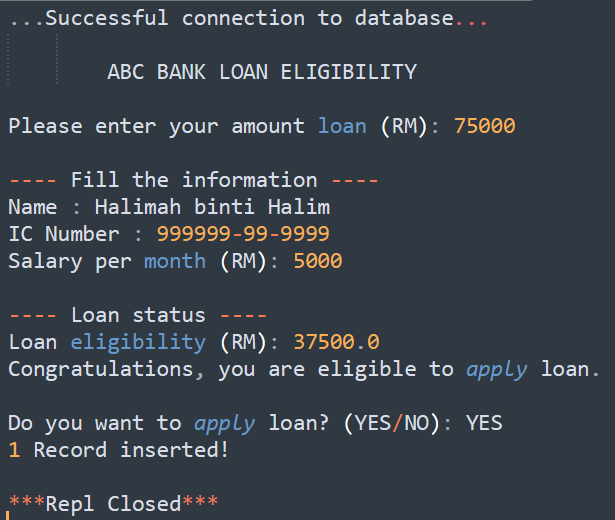
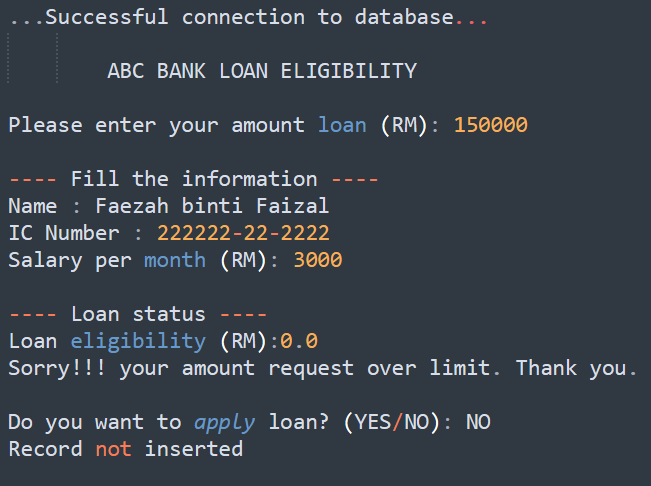
 

Figure 1: Eligible Status Figure 2: Not Eligible Status

**QUESTION:**

1. Write and run code at Figure3 to file named connection.py.

|  |
| --- |
| **import mysql.connector**  **mydb = mysql.connector.connect(**  **host="localhost",**  **user="root",**  **password="")**  **mycursor = mydb.cursor()**  **mycursor.execute("CREATE DATABASE loan")**  **dataBase = mysql.connector.connect(**  **host = "localhost",**  **user = "root",**  **password = "",**  **database = "loan")**  **cursorObj = dataBase.cursor()**  **loanB = """CREATE TABLE listloan (**  **icnumber VARCHAR(20) NOT NULL,**  **name VARCHAR(50),**  **salary VARCHAR(50),**  **amount VARCHAR(12),**  **loanStatus VARCHAR(12),**  **PRIMARY KEY (icnumber))"""**  **cursorObj.execute(loanB)**  **dataBase.close()** |

Figure 3: Database code

1. Manipulates the given code at Figure3 to make it as module for database connection only. This module contains one class named as **connectionDB** and magic method **\_\_init\_\_** for connection between program and database.
2. Build one program to solve the given scenario. Use file main.py to write main program to produce output like Figure1.
3. Create check\_eligibility() function to check loan eligibility based on the given requirement.
4. Create insert() function to add data in database when the borrower agrees to apply loan.
5. Perform the exception handling to handle process add data into database. Use message box like Figure 4 to inform user.

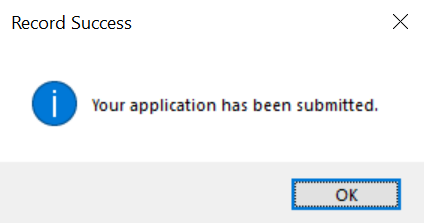
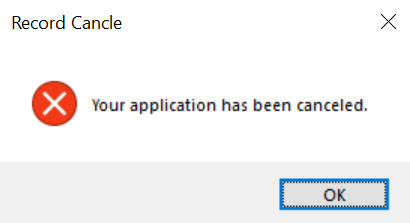
 

Figure 4: Message box

1. Write code to display as prove these two files at your registration number folder like a Figure 5.

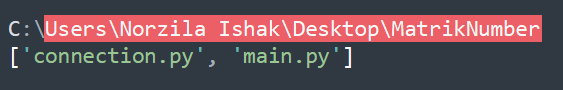


Figure 5: Directory location

**EXAMPLE 3:**

1. Your responsibility as a programmer employed at Lelemove, a shipping company, is to create an application capable of determining the cost of parcels based on their dimensions. During your investigation of the company's code repositories, you came across the previous programmer's main program (main.py). Your objective is to finalize the program, enhancing Lelemove's software for calculating parcel prices. The pricing table for parcels is outlined below:

| **Weight (kg)** | **Volume (cm³)** | **Price ($)** |
| --- | --- | --- |
| 0-1 | 0-5000 | 3 |
| 0-1 | 5001-10,000 | 5 |
| 0-1 | >10,000 | 7 |
| 1-5 | 0-5000 | 5 |
| 1-5 | 5001-10,000 | 7 |
| 1-5 | >10,000 | 9 |
| >5 | 0-5000 | 7 |
| >5 | 5001-10,000 | 9 |
| >5 | >10,000 | 11 |

Formula to calculate volume is

Volume = length x width x height

|  |
| --- |
| from parcel\_calculator import calculate\_price  # Get user input for parcel dimensions and weight  length = float(input("Enter parcel length in centimeters: "))  width = float(input("Enter parcel width in centimeters: "))  height = float(input("Enter parcel height in centimeters: "))  weight = float(input("Enter parcel weight in kilograms: "))  # Calculate parcel price using imported function  price = calculate\_price(length, width, height, weight)  # Print parcel price to user  print("The price of your parcel is: $", price) |
| ***main.py*** |

Read the question below to guide you to complete the program.

1. Build the *main.py* program. Do you encounter any error? (give screenshot)
2. Describe the meaning of this error?
3. Constructs another module to complete the *main.py* program and save length,width,height and weight of parcel on text file.
4. During your exploration of the company's code repositories, you stumbled upon another program that remains unfinished. This time, your objective is to develop a Graphical User Interface (GUI) for the program, enabling it to calculate parcel prices. Apply your programming expertise and understanding to finalize the program.

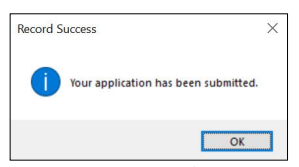
|  |
| --- |
| Import x as tk  import parcel\_calculator  class x :  def \_\_init\_\_(self, master):  self.master = master  master.title("Parcel Calculator")  # Create length input label and entry field  self.length\_label = tk.Label(master, text="Length (cm):")  self.length\_label.grid(row=0, column=0)  self.length\_entry = tk.Entry(master)  self.length\_entry.grid(row=0, column=1)  # Create width input label and entry field  self.width\_label = tk.Label(master, text="Width (cm):")  self.width\_label.grid(row=1, column=0)  self.width\_entry = tk.Entry(master)  self.width\_entry.grid(row=1, column=1)  # Create height input label and entry field  self.height\_label = tk.Label(master, text=" x :")  self.height\_label.grid(row=2, column=0)  self.height\_entry = tk.Entry(master)  self.height\_entry.grid(row=2, column=1)  # Create weight input label and entry field  self.weight\_label = tk.Label(master, text="Weight (kg):")  self.weight\_label.grid(row=3, column=0)  self.weight\_entry = tk.Entry(master)  self.weight\_entry.grid(row=3, column=1)  # Create calculate button  self.calculate\_button = tk.Button(master, text="Calculate", command=self. Cx )  self.calculate\_button.grid(row=4, column=0, columnspan=2)  # Create price label  self.price\_label = tk.Label(master, text="")  self.price\_label.grid(row=5, column=0, columnspan=2)  def calculate\_price(self):  # Get parcel dimensions and weight from entry fields  length = float(self.length\_entry.get())  width = float(self.width\_entry.get())  height = float(self.height\_entry.get())  weight = float(self.weight\_entry.get())    # Calculate parcel price using imported function  price = parcel\_calculator.calculate\_price(l x , x , x , x )  # Update price label with calculated price  self.price\_label.config(text="The price of your parcel is: $" + str( x ))  x = tk.Tk()  parcel\_calculator\_gui = ParcelCalculatorGUI(a)  a.mainloop() |
| ***gui\_parcel.py*** |

1. Manipulates x with your answer to complete the program above to get a GUI version of the program.
2. Upon completing the program, you noticed that the graphical user interface (GUI) lacked visual appeal, potentially hindering user experience. Your assignment is to construct at least 3 additional GUI features to create a more user-friendly interface.

i.e additional message box, pull up button, color variations, layout and etc.

1. The development of the parcel shipping calculation program is nearing completion. The remaining task involves implementing a feature to store all shipment data in a database, utilizing any TinySQL/SQL system of your choice. To accomplish this, your responsibility is to create an additional program specifically designed to capture and record the relevant data.
2. Builds a database named Lelemove System.
3. Make a table with given attributes as in table below with item\_id as the primary key.

|  |  |
| --- | --- |
| **Attribute** | **Data type** |
| Item\_id | Varchar(10) |
| Item\_height | Varchar(5) |
| Item\_width | Varchar(5) |
| Item\_length | Varchar(5) |
| Item\_volume | Varchar(10) |
| Item\_price | Varchar(5) |

1. Constructs a button to add all the data given by the user to the database.
2. Perform the exception handling (message box) to notify the user that the data given has been stored.

i.e

**EXAMPLE 4:**

**QUESTION 1**

Complete the program below based on this question.

Build a class called Time whose only field is a time in seconds. It should have a method called convert\_to\_minutes that returns a string of minutes and seconds formatted as in the following example: if seconds is 3680, the method should return '61:20'. It should also have a method called convert\_to\_hours that returns a string of hours, minutes, and seconds formatted analogously to the previous method.

Example:

# Convert to Minutes

3680 seconds = 61:20 minutes (61 Minutes and 20 seconds)

# Convert to Hours

3680 seconds = 1:1:20 hour (1 hour 1 minutes 20 seconds)

Tips:

1 hour = 3600 seconds

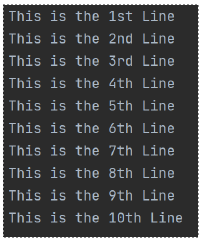
1 minutes = 60 seconds

**QUESTION 2**

Write a python program that create a text file called myLine.txt as shown in Figure 1 and do the followings.

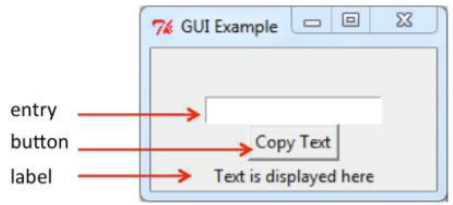
b) A Python program which allows you to read the first 5 lines of myLine.txt.

c) A program to read the last 5 lines of myLine.txt



**QUESTION 3**

Create a GUI as shown in Figure.



Program flow:

When user entered text in the entry box, the text will appear in the label text when the button is clicked.