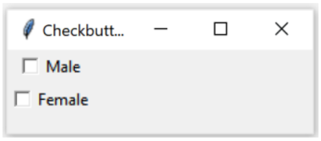
Part – A (10 x 1 = 10 Marks)						
Q. No	**Question**	**Marks**	**BL**	**CO**	**PO**	**P.I**
1	Which if the following property of Geometry manager pack allows the widget to fill any space not otherwise used in widget's parent? **a) Fill** b) span c) expand d) pad	**1**	**2**	**2**	**2**	**3.8.2**
2	Config() in Python Tkinter are used for ___ a. place the widget b. destroy the widget c. configure the widget **d. change property of the widget**	**1**	**2**	**2**	**4**	**3.8.2**
3	Correct way to run the query in python sqlite3 is : a) sqlite3.execute.query **b) sqlite3.execute(query)** c) sqlite.run.execute(query) d) sql.run.execute(query)	**1**	**2**	**3**	**3**	**3.8.2**
4	How can you delete all of the rows where the "name" is "Ruby" in the Cats Table? a. **DELETE FROM Cats WHERE name = 'Ruby'** b. DELETE name='Ruby' FROM Cats c. DELETE ROW name='Ruby' FROM Cats d. DELETE FROM Cats WHERE name == 'Ruby'	**1**	**3**	**3**	**3**	**3.8.2**
5	What is the method that wakes up all thread waiting for the condition? **a. notifyAll()** b. release() c. releaseAll() d. notify()	**1**	**1**	**2**	**2**	**3.8.2**
6	What does the Thread.join() method do? (a) Adds the thread to a pool (b) Merges two threads into one (c) **Waits for the thread to finish** (d) Restricted access to a resource	**1**	**2**	**2**	**4**	**3.8.2**
7	How pack() functions works on tkinter widget? a) According to x,y coordinate b) According to row and column wise c) **According to left, right, up and down** d) Both a and b	**1**	**2**	**2**	**2**	**3.8.2**
8	Which of the following property of Geometry manager pack allows the widget to fill any space not otherwise used in widget's parent? a. fill b. span **c.expand** d. pad	**1**	**1**	**3**	**3**	**3.8.2**
9	"Consider the following output.#variable "window refers to the main window created using tk". What is the syntax for creating the radiobutton with caption "C"? R1 = Radiobutton(window, caption="C", variable=radio, value=1) R1 = Radiobutton(top, text="C", variable=radio, value=1) **R1 = Radiobutton(window, text="C", variable=radio, value=1)** R1 = Radiobutton(top, caption="C", variable=radio, value=1)	**1**	**2**	**2**	**2**	**3.8.2**
10	Which of the following method used to terminate the running thread? a) join() b) wait() **c) release()** d) notify()	**1**	**2**	**2**	**2**	**3.8.2**
Part – B (5 x 4 = 20 Marks) Instructions: Answer all Five Questions						
6	Explain about the options available with pack() method of Geometry manager. fill The default value of this option is set to NONE. Also, we can set it to X or Y in order to determine whether the widget contains any extra space. side This option specifies which side to pack the widget against. If you want to pack widgets vertically, use TOP which is the default value. If you want to pack widgets horizontally, use LEFT. expand This option is used to specify whether the widgets should be expanded to fill any extra space in the geometry master or not. Its default value is false. If it is false then the widget is not expanded otherwise widget expands to fill extra space.	**4**	**3**	**3**	**3**	**3.8.2**

7	Write a python program to sum the elements of list using declarative and imperative style.
	def sum(l):
	    s=0
	    for i in l:
	        s+=i
	    return s
	
	l =[23,54,65,87,12,78]
	print(sum(l))
		4	**2**	**2**	**2**	**3.8.2**

8	Write a Python code using tkinter for below window (Condition : when you click 1st checkbutton, print ‘MALE’. Similarly, when you click 2nd checkbutton, print ‘FEMALE’. 	**4**	**2**	**2**	**3**	**3.8.2**
	cb1= IntVar()					
	b1 = Checkbutton(root, text = "Male",variable = cb1,onvalue = 1,offvalue = 0,cmd=test)					
	b2 = Checkbutton(root, text = "Female",variable = cb1,onvalue = 3,offvalue = 2,cmd=test)					
	def test():					
	if cb1==1:					
	messagebox.showinfo("Gender","Male")					
	elif cb1==3:					
	messagebox.showinfo("Gender","Female")					
9	Draw a neat diagram and explain Implicit Parallelism	**4**	**2**	**3**	**4**	**3.8.2**
	Implicit Parallelism allows programmers to write their programs without any concern about the exploitation of parallelism. Exploitation of parallelism is instead automatically performed by the compiler and/or the runtime system. In this way the parallelism is transparent to the programmer (except, hopefully, for an improvement in execution performance), maintaining the complexity of software development at the same level of standard sequential programming.					
10	Write a python code to create Mythread by extending the Thread class and implement the behaviour of thread to sort the given set of number and print the sorted list.	**4**	**2**	**2**	**3**	**3.8.2**
	def srt(l):					
	l.sort()					
	print(l)					
	num = [11, 3, 7, 5, 2]					
	t1 = threading.Thread(target=print_square, args=(num,))					
	t1.start()					
	t1.join()					
	Part – C (2 x 10 = 20 Marks) Instructions: Answer for two questions					
11 a	Write a python sqlite to create a table named product with product id, product name, price, manufacturer and quantity and Insert samples rows into the tables and perform the following.	**10**	**3**	**3**	**3**	**3.8.2**
	a. Find the three most expensive products					
	b. Find the Products manufactured by a "ABC" company with a price less than 200					
	c. What products have a price between $20 and $200					
11b	d. How many products are in the database?					
	e. What product categories do we have?	**10**	**3**	**3**	**3**	
	Import sqlite3					
	Con=sqlite3.connect(“prod.db”)					
	st = “create table product(name char(50), price number(5), category char(50), manuf char(50));”					
	Con.execute(st)					
	st=”insert into product values ('laptop' ,100000,’gadgets’,’hp’);”					
	Con.execute(st)					
	st=”SELECT Name, Price FROM Product ORDER BY Price DESC, Name ASC LIMIT 0,3;”					
	Con.execute(st);					
	st =”SELECT Name, Price FROM Product where manufacture='ABC' and price<200;”					
	con.execute(st);					
	st= “SELECT Name, Price FROM Product where price between 20 and 200;”					

```
con.execute(st)
st= “Select count(distinct name) from product;”
con.execute(st)
st= “select distinct(pcategories) from product”
con.execute(st)
con.close()
```

(OR)

John needs to create a Bank Class with initial balance of 10000 who’s withdrawn and deposit methods are synchronized. Supply one thread, which keeps withdrawing 3000 from the account for 5 times when there is no sufficient funds it has to wait. Similarly, the second thread keeps depositing amount of 2000 into the account for 5 times. Ensure both threads run in tandem without conflicts. Help john to implement the above scenario using python multithreading concept without race condition.

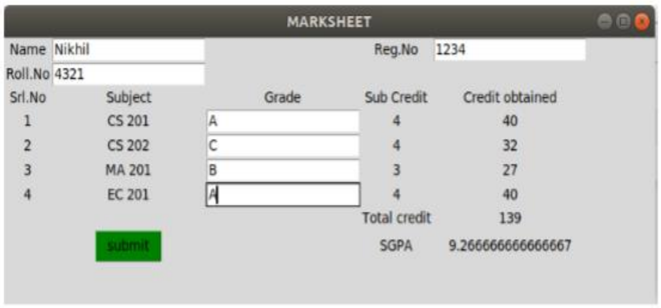
Ans:

```
class Bank_Account:
        lock = threading.Lock()
        def __init__(self):
                self.balance=0
        def deposit(self):
                lock.acquire()
                amount=float(input("Enter amount to be Deposited: "))
                self.balance += amount
                print("\n Amount Deposited:",amount)
                lock.release()

        def withdraw(self):
                lock.acquire()
                amount = float(input("Enter amount to be Withdrawn: "))
                if self.balance>=amount:
                        self.balance-=amount
                        print("\n You Withdrew:", amount)
                else:
                        print("\n Insufficient balance ")
                lock.release()

        def display(self):
                print("\n Net Available Balance=",self.balance)

s = Bank_Account()

t1 = threading.Thread(target=s.deposit(), args=(,))
t2 = threading.Thread(target=s.withdraw(), args=(,))
t3 = threading.Thread(target=s.deposit(), args=(,))
t4 = threading.Thread(target=s.withdraw(), args=(,))
t5 = threading.Thread(target=s.deposit(), args=(,))
t1.start()
t1.join()
t2.start()
t2.join()
t3.start()
t3.join()
t4.start()
t4.join()
t5.start()
t5.join()
```

12 a	Write a tkinter code to create simple form to capture the details for address book entry and display the information using message box (OR)	**10**	**3**	**2**	**3**	**3.8.2**
12 b		**10**	**3**	**2**	**3**	

```
# Import Module
from tkinter import *

# Create Object
root = Tk()

# Set geometry
root.geometry('400x500')

# Add Buttons, Label, ListBox
Name = StringVar()
Number = StringVar()

frame = Frame()
frame.pack(pady=10)

frame1 = Frame()
frame1.pack()

frame2 = Frame()
frame2.pack(pady=10)

Label(frame, text = 'Name', font='arial 12 bold').pack(side=LEFT)
Entry(frame, textvariable = Name,width=50).pack()

Label(frame1, text = 'Phone No.', font='arial 12 bold').pack(side=LEFT)
Entry(frame1, textvariable = Number,width=50).pack()

Label(frame2, text = 'Address', font='arial 12 bold').pack(side=LEFT)
address = Text(frame2,width=37,height=10)
address.pack()

Button(root,text="Add",font="arial 12 bold").place(x= 100, y=270)
Button(root,text="View",font="arial 12 bold").place(x= 100, y=310)
Button(root,text="Delete",font="arial 12 bold").place(x= 100, y=350)
Button(root,text="Reset",font="arial 12 bold").place(x= 100, y=390)

scroll_bar = Scrollbar(root, orient=VERTICAL)
select = Listbox(root, yscrollcommand=scroll_bar.set, height=12)
scroll_bar.config (command=select.yview)
scroll_bar.pack(side=RIGHT, fill=Y)
select.place(x=200,y=260)

# Execute Tkinter
root.mainloop()
```

Write a tkinter code to design the given application.



MARKSHEET

Name Nikhil Reg.No 1234
Roll.No 4321

Srl.No	Subject	Grade	Sub Credit	Credit obtained
1	CS 201	A	4	40
2	CS 202	C	4	32
3	MA 201	B	3	27
4	EC 201	A	4	40
			Total credit	139
			SGPA	9.266666666666667

Ans:

```
import tkinter as tk
master = tk.Tk()
master.title("MARKSHEET")
master.geometry("700x250")
e1 = tk.Entry(master)
e2 = tk.Entry(master)
e3 = tk.Entry(master)
e4 = tk.Entry(master)
e5 = tk.Entry(master)
e6 = tk.Entry(master)
e7 = tk.Entry(master)

def display():

        tot = 0
        if e4.get() == "A":
                tk.Label(master, text="40").grid(row=3, column=4)
                tot += 40
        if e4.get() == "B":
                tk.Label(master, text="36").grid(row=3, column=4)
                tot += 36
        if e4.get() == "C":
                tk.Label(master, text="32").grid(row=3, column=4)
                tot += 32
        if e4.get() == "D":
                tk.Label(master, text="28").grid(row=3, column=4)
                tot += 28
        if e4.get() == "P":
                tk.Label(master, text="24").grid(row=3, column=4)
                tot += 24

        if e4.get() == "F":
                tk.Label(master, text="0").grid(row=3, column=4)
                tot += 0

        # Similarly doing with other objects
        if e5.get() == "A":
                tk.Label(master, text="40").grid(row=4, column=4)
                tot += 40
        if e5.get() == "B":
                tk.Label(master, text="36").grid(row=4, column=4)
                tot += 36
        if e5.get() == "C":
                tk.Label(master, text="32").grid(row=4, column=4)
                tot += 32
        if e5.get() == "D":
                tk.Label(master, text="28").grid(row=4, column=4)
                tot += 28
        if e5.get() == "P":
                tk.Label(master, text="28").grid(row=4, column=4)
                tot += 24
        if e5.get() == "F":
                tk.Label(master, text="0").grid(row=4, column=4)
                tot += 0

        if e6.get() == "A":
```

```
        tk.Label(master, text="30").grid(row=5, column=4)
        tot += 30
    if e6.get() == "B":
        tk.Label(master, text="27").grid(row=5, column=4)
        tot += 27
    if e6.get() == "C":
        tk.Label(master, text="24").grid(row=5, column=4)
        tot += 24
    if e6.get() == "D":
        tk.Label(master, text="21").grid(row=5, column=4)
        tot += 21
    if e6.get() == "P":
        tk.Label(master, text="28").grid(row=5, column=4)
        tot += 24
    if e6.get() == "F":
        tk.Label(master, text="0").grid(row=5, column=4)
        tot += 0

    if e7.get() == "A":
        tk.Label(master, text="40").grid(row=6, column=4)
        tot += 40
    if e7.get() == "B":
        tk.Label(master, text="36").grid(row=6, column=4)
        tot += 36
    if e7.get() == "C":
        tk.Label(master, text="32").grid(row=6, column=4)
        tot += 32
    if e7.get() == "D":
        tk.Label(master, text="28").grid(row=6, column=4)
        tot += 28
    if e7.get() == "P":
        tk.Label(master, text="28").grid(row=6, column=4)
        tot += 24
    if e7.get() == "F":
        tk.Label(master, text="0").grid(row=6, column=4)
        tot += 0

    # to display total credits
    tk.Label(master, text=str(tot)).grid(row=7, column=4)

    # to display SGPA
    tk.Label(master, text=str(tot/15)).grid(row=8, column=4)

# label to enter name
tk.Label(master, text="Name").grid(row=0, column=0)

# label for registration number
tk.Label(master, text="Reg.No").grid(row=0, column=3)

# label for roll Number
tk.Label(master, text="Roll.No").grid(row=1, column=0)

# labels for serial numbers
tk.Label(master, text="Srl.No").grid(row=2, column=0)
tk.Label(master, text="1").grid(row=3, column=0)
tk.Label(master, text="2").grid(row=4, column=0)
tk.Label(master, text="3").grid(row=5, column=0)
tk.Label(master, text="4").grid(row=6, column=0)



# labels for subject codes
tk.Label(master, text="Subject").grid(row=2, column=1)
tk.Label(master, text="CS 201").grid(row=3, column=1)
tk.Label(master, text="CS 202").grid(row=4, column=1)
tk.Label(master, text="MA 201").grid(row=5, column=1)
tk.Label(master, text="EC 201").grid(row=6, column=1)
```

```
# label for grades
tk.Label(master, text="Grade").grid(row=2, column=2)
e4.grid(row=3, column=2)
e5.grid(row=4, column=2)
e6.grid(row=5, column=2)
e7.grid(row=6, column=2)


# labels for subject credits
tk.Label(master, text="Sub Credit").grid(row=2, column=3)
tk.Label(master, text="4").grid(row=3, column=3)
tk.Label(master, text="4").grid(row=4, column=3)
tk.Label(master, text="3").grid(row=5, column=3)
tk.Label(master, text="4").grid(row=6, column=3)

tk.Label(master, text="Credit obtained").grid(row=2, column=4)

# taking entries of name, reg, roll number respectively
e1 = tk.Entry(master)
e2 = tk.Entry(master)
e3 = tk.Entry(master)

# organizing them in the grid
e1.grid(row=0, column=1)
e2.grid(row=0, column=4)
e3.grid(row=1, column=1)

# button to display all the calculated credit scores and sgpa
button1 = tk.Button(master, text="submit", bg="green", command=display)
button1.grid(row=8, column=1)


tk.Label(master, text="Total credit").grid(row=7, column=3)
tk.Label(master, text="SGPA").grid(row=8, column=3)


master.mainloop()
```