B1 - SET - A

PART A 10 X 1 = 10 Marks

1. How pack() function 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

2. what is the correct syntax of destroy in tkinter?

   a. destroy(object)
   b. object.destroy()
   c. object(destroy)
   d. delete(object)

3. By using sqlite3, we can store our data on

   a. local
   b. global
   c. server
   d. both server and global

4. connect() function is Sqlite2 is used for?

   a. to connect the database

   b. to open the database

   c. to create a database

   d. both a, b and c

5. How does run() method is invoked?

   a. By Thread.run()

   B . By Thread.start()

   C . By Thread.create()

   D . By Thread.join()

6. Which thread method is used to wait until it terminates?

   A . waitforthread()
   B . join()

   C . None

   D . wait()

7. Race conditions are …
   (a) Two threads incorrectly accessing a shared resource
   (b) Testing which thread completes first
   (c) Something you should add to your code
   (d) The weather on race day

8. ----------is needed to prevent mutual exclusion while accessing shared resources

   a. Process
   b. thread
   c. Synchronization
   d. race condition

9. Predict the output of the following program

```
L = [lambda x: x ** 2,
     lambda x: x ** 3,
     lambda x: x ** 4]

for f in L:
    print(f(3))
```

   a. 9 27 81
   b. 27 9 81
   c. 9 81 27
   d. 81 9 27

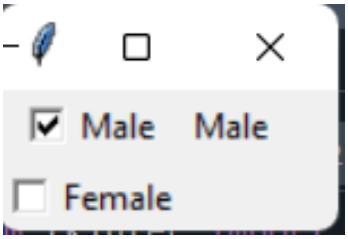
10. What will be the output of the following Python code?

```
from functools import reduce
def do_sum(x1, x2):
    return x1 + x2
print(reduce(do_sum, [1, 2, 3, 4]))
```

   a. 13
   b. 11
   c. 12
   d. 10

11)

```
from tkinter import *
r=Tk()
def func1():
   label1.configure(text="Male")
def func2():
   label2.configure(text="Female")
r.title("Checkbutton")
c1=Checkbutton(r,text="Male",command=func1)
c1.grid(row=0,column=0)
c2=Checkbutton(r,text="Female",command=func2)
c2.grid(row=1,column=0)
label1=Label(r,text="")
label1.grid(row=0,column=1)
label2=Label(r,text="")
label2.grid(row=1,column=1)
r.mainloop()
```



12)

Imperative Programming	Declarative Programming
In this, programs specify how it is to be done.	In this, programs specify what is to be done.
It simply describes the control flow of computation.	It simply expresses the logic of computation.
Its main goal is to describe how to get it or accomplish it.	Its main goal is to describe the desired result without direct dictation on how to get it.
Its advantages include ease to learn and read, the notional model is simple to understand, etc.	Its advantages include effective code, which can be applied by using ways, easy extension, high level of abstraction, etc.
Its type includes procedural programming, object-oriented programming, parallel processing approach.	Its type includes logic programming and functional programming.

```
#Declarative example : -

small_nums = [x for x in range(20) if x<5]
print(small_nums)
```

```
[0, 1, 2, 3, 4]
```

```
#Imperative example:-

small_nums=[]
for i in range(20):
    if i<5:
        small_nums.append(i)
print(small_nums)
```

```
[0, 1, 2, 3, 4]
```

13)

```
import threading
def func1():
    for i in range(5):
        print(i*i)
def func2():
    for i in range(5):
        print(i*i*i)
t1=threading.Thread(target=func1)
t2=threading.Thread(target=func2)
t1.start()
t2.start()
t1.join()
t2.join()
```

```
0
1
4
9
16
0
1
8
27
64
```

14)

```
import threading
x = 0    # A shared value
COUNT = 100
def incr():
    global x
    for i in range(COUNT):
        x += 1
        print(x)
def decr():
    global x
    for i in range(COUNT):
        x -= 1
        print(x)
t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
print(x)
```

15)

```
texts = ["php", "w3r", "Python", "abcd", "Java", "aaa"]
print("Orginal list of strings:")
print(texts)
result = list(filter(lambda x: (x == "".join(reversed(x))), texts))
print("\nList of palindromes:")
print(result)
```

```
Orginal list of strings:
['php', 'w3r', 'Python', 'abcd', 'Java', 'aaa']

List of palindromes:
['php', 'aaa']
```

16)

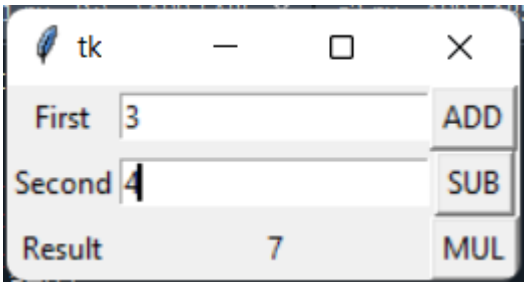
```
import sqlite3
conn=sqlite3.connect('App.db')
print("Opened database successfully")
conn.execute("CREATE TABLE Student(SNo INT PRIMARY KEY NOT NULL,Regno text,name text,gender text,email text,phone text,dept text);")
print("Table created successfully")
conn.execute("INSERT INTO Student VALUES(1,490,'Someone','male','xyz@gmail.com','098172','Ctech');")
conn.execute("INSERT INTO Student VALUES(2,800,'Anyone','male','pqr@gmail.com','986643','ECE');")
conn.execute("INSERT INTO Student VALUES(3,360,'Random','female','lmn@gmail.com','875567','Biotech');")
conn.execute("INSERT INTO Student VALUES(4,440,'StudRa','female','dgf@gmail.com','765357','Ctceh');")
conn.commit()
print("Records inserted successfully");
print("\nDisplaying the table \n")
for rows in conn.execute("SELECT * from Student;"):
   print(rows)
print("\nPART B\n")
for rows in conn.execute("SELECT regno,name,dept from Student;"):
   print(rows)
print("\nPART C\n")
for rows in conn.execute("SELECT name from Student WHERE name LIKE '%Ra';"):
   print(rows)
print("\nPART D\n")
for rows in conn.execute("SELECT name from Student WHERE gender='female';"):
   print(rows)
print("\nPART E\n")
for rows in conn.execute("SELECT name from Student ORDER BY name DESC"):
   print(rows)
conn.close()
```

```
Opened database successfully
Table created successfully
Records inserted successfully

Displaying the table

(1, '490', 'Someone', 'male', 'xyz@gmail.com', '098172', 'Ctech')
(2, '800', 'Anyone', 'male', 'pqr@gmail.com', '986643', 'ECE')
(3, '360', 'Random', 'female', 'lmn@gmail.com', '875567', 'Biotech')
(4, '440', 'StudRa', 'female', 'dgf@gmail.com', '765357', 'Ctceh')

PART B

('490', 'Someone', 'Ctech')
('800', 'Anyone', 'ECE')
('360', 'Random', 'Biotech')
('440', 'StudRa', 'Ctceh')

PART C

('StudRa',)

PART D

('Random',)
('StudRa',)
```

```
PART E

('StudRa',)
('Someone',)
('Random',)
('Anyone',)
```

17)

```
from tkinter import *
r=Tk()
def add():
   a=int(entry1.get())
   b=int(entry2.get())
   c=str(a+b)
   label4.configure(text=c)
def sub():
   a=int(entry1.get())
   b=int(entry2.get())
   c=str(a-b)
   label4.configure(text=c)
def  mul():
   a=int(entry1.get())
   b=int(entry2.get())
   c=str(a*b)
   label4.configure(text=c)
label1=Label(r,text="First")
label2=Label(r,text="Second")
label3=Label(r,text="Result")
label4=Label(r,text="")
```



```
entry1=Entry(r)
entry2=Entry(r)
button1=Button(r,text="ADD",command=add)
button2=Button(r,text="SUB",command=sub)
button3=Button(r,text="MUL",command=mul)
label1.grid(row=0,column=0)
label2.grid(row=1,column=0)
label3.grid(row=2,column=0)
label4.grid(row=2,column=1)
entry1.grid(row=0,column=1)
entry2.grid(row=1,column=1)
button1.grid(row=0,column=2)
button2.grid(row=1,column=2)
button3.grid(row=2,column=2)
r.mainloop()
```

18)

```
import threading
lock=threading.Lock()
def calc_square(numbers):
    lock.acquire()
    print("Calculate square numbers")
    for n in numbers:
        print('square : ', n*n)
    lock.release()
def calc_cube(numbers):
    lock.acquire()
    print("Calculate cube of numbers")
    for n in numbers:
        print('cube : ', n*n*n)
    lock.release()
arr = [2,3,8,9]
t1 = threading.Thread(target=calc_square, args=(arr,))
t2 = threading.Thread(target=calc_cube, args=(arr,))
t1.start()
t2.start()
t1.join()
t2.join()
```

```
Calculate square numbers
square :  4
square :  9
square :  64
square :  81
Calculate cube of numbers
cube :  8
cube :  27
cube :  512
cube :  729
```

19)

a)

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("Original list of integers:")
print(nums)
print("\nSquare every number of the said list:")
square_nums = list(map(lambda x: x ** 2, nums))
print(square_nums)
print("\nCube every number of the said list:")
cube_nums = list(map(lambda x: x ** 3, nums))
print(cube_nums)
```

```
Original list of integers:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Square every number of the said list:
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

Cube every number of the said list:
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

b)

```
alphabets = ['a', 'b', 'U', 'd', 'e', 'i', 'j', 'o', 'x', 'A', 'Z', 't']
def filterVowels(alpha):
    vowels = ['a', 'e', 'i', 'o', 'u']
    if(alpha.lower() in vowels):
        return True
    else:
        return False
filter_vowel = filter(filterVowels, alphabets)
print("Original Sequence :",alphabets)
print("\nFiltered letters are : ")
for i in filter_vowel:
    print(i,end=" ")
```