18CSC207J – Advanced Programming Practice

Answer Key

B2-Set A
CT2

11.

There are three built-in methods to position the widgets in the application frame or the screen in tkinter:
a. pack
b. grid
c. place

Pack Method:
Pack method is used to organize the widgets in horizontal and vertical boxes.
Options/Attributes Used in Pack method:

a. Fill: Fill packs a widget inside a container, filling the entire container.
b. Padx: Padx option is used to pad the widget externally along the x-axis, (left and right)
c. Pady: Pady option is used to pad the widget externally along the y-axis, (top and bottom)
d. Side: Decides the positioning widgets (left, right, top, bottom).
Expand, ipadx, ipady, etc. are the other options used in pack method.

12.

```
from tkinter import *
master = Tk()

master.configure(background="Green")
def setColor():
    c = (color.get())
    master.configure(background = c)

color = StringVar()
Entry(master, textvariable=color).pack()
Button(master, text="SET", command=setColor).pack()

master.mainloop()
```

13.

Functional and Procedural Programming:
Functional Programming: A functional Programming language uses functions to execute the program throughout the code, i.e. a function that takes *n* arguments and returns a value.
e.g.: - JavaScript, OCaml, Scala.

Procedural Programming: A procedural language, on the other hand, performs a series of *sequential* steps. They are procedure oriented programming languages. They follow series of operations to be performed.

- Everything is done in a specific order.
- Execution of a routine may have side effects.

e.g. C, C++, are the most common procedural programming languages.

14.

```
# Printing Even and Odd Number from given list
listVal = [2, 5, 8, 23, 14, 37, 47, 18, 20, 36, 29]
print("Even numbers are: ")
for i in listVal:
    if i%2 == 0:
        print(i, end = " ")
print("\nOdd numbers are: ")
for i in listVal:
    if i%2 != 0:
        print(i, end = " ")
```

15.

```
import threading
listItems = [5, 8, 3, 9, 6]
def sortArr():
    global listItems
    listItems = sorted(listItems)
    for i in listItems:
        print(i, end=" ")
Mythread = threading.Thread(target=sortArr)
Mythread.start()
```

17.

```
# tkinter Program
from tkinter import *
master = Tk()
master.title("MARKSHEET")

# First Row
Label(master, text="Name", width=10).grid(row=0)
Entry(master, width=30).grid(row=0, column=1)
Label(master, text="Reg No", width=10).grid(row=0, column = 3)
Entry(master, width=30).grid(row=0, column=4)

# Second Row
Label(master, text="Roll No", width=10).grid(row=1)
Entry(master, width=30).grid(row=1, column=1)

# Third Row
Label(master, text="Sr. No.", width=10).grid(row=2)
Label(master, text="Subject", width=30).grid(row=2, column=1)
Label(master, text="Graph", width=30).grid(row=2, column=2)
Label(master, text="Sub Credit", width=10).grid(row=2, column = 3)
Label(master, text="Credit Obtained",width=30).grid(row=2, column=4)

# Fourth Row
Label(master, text="1", width=10).grid(row=3)
Label(master, text="CS 201", width=30).grid(row=3, column=1)
Entry(master, width=30).grid(row=3, column=2)
Label(master, text="4", width=10).grid(row=3, column = 3)
Label(master, text="40",width=30).grid(row=3, column=4)

# Fifth Row
Label(master, text="2", width=10).grid(row=4)
Label(master, text="CS 202", width=30).grid(row=4, column=1)
Entry(master, width=30).grid(row=4, column=2)
Label(master, text="3", width=10).grid(row=4, column = 3)
Label(master, text="32",width=30).grid(row=4, column=4)

# Sixth Row
Label(master, text="3", width=10).grid(row=5)
Label(master, text="MA 201", width=30).grid(row=5, column=1)
Entry(master, width=30).grid(row=5, column=2)
Label(master, text="4", width=10).grid(row=5, column = 3)
Label(master, text="27",width=30).grid(row=5, column=4)

# Seventh Row
Label(master, text="4", width=10).grid(row=6)
Label(master, text="EC 201", width=30).grid(row=6, column=1)
Entry(master, width=30).grid(row=6, column=2)
Label(master, text="4", width=10).grid(row=6, column = 3)
Label(master, text="40",width=30).grid(row=6, column=4)

# Seventh Row
Label(master, text="Total Credit", width=10).grid(row=7, column = 3)
Label(master, text="139",width=30).grid(row=7, column=4)

# Eighth Row
Button(master, text="Calculate").grid(row=8, column =1)
Label(master, text="SGPA", width=10).grid(row=8, column = 3)
Label(master, text="9.266666666666667",width=30).grid(row=8, column=4)

master.mainloop()
```

18.

Lazy Evaluation: Lazy evaluation is an evaluation strategy which holds the evaluation of an expression until its value is needed. It avoids repeated evaluation. It’s a functional programming paradigm

Lazy Evaluation – Advantages

- It allows the language runtime to discard sub-expressions that are not directly linked to the final result of the expression.
- It reduces the time complexity of an algorithm by discarding the temporary computations and conditionals.

- It allows the programmer to access components of data structures out-of-order after initializing them, as long as they are free from any circular dependencies.
- It is best suited for loading data which will be infrequently accessed.

Lazy Evaluation − Drawbacks

- It forces the language runtime to hold the evaluation of sub-expressions until it is required in the final result by creating **thunks** (delayed objects).
- Sometimes it increases space complexity of an algorithm.
- It is very difficult to find its performance because it contains thunks of expressions before their execution.

Suitable Example :

```
r = range(10)
print(r)
range(0, 10)
print(r[3])
```

Closure: A closure is a function (object) that remembers its creation environment (enclosing scope).

Example:

```
def closureFunc(start):
  def incrementBy(inc):
    return start + inc
  return incrementBy
closure1 = closureFunc(9)
closure2 = closureFunc(90)
print ('clsure1(3) = %s' %(closure1(3)))
print ('closure2(3) = %s' %(closure2(3)))
```

19.

```
import sqlite3
conn = sqlite3.connect('testdb.db')
conn.execute('''create table product (
                productId INT PRIMARY KEY NOT NULL,
                productName TEXT NOT NULL,
                PRICE REAL NOT NULL,
                MANUFACTURER TEXT NOT NULL,
                QUANTITY INT NOT NULL,
                CATEG TEXT NOT NULL);''')

listItems = [
    (1, 'Apple', 200, 'ABC', 20, 'Fruit'),
    (2, 'Orange', 100, 'ABD', 10, 'Fruit'),
    (3, 'Butterflow', 10, 'AAA', 150, 'Pen')
]

conn.executemany("INSERT INTO product VALUES (?, ?, ?, ?, ?, ?)",
listItems)

conn.commit()
print('Database created successfully')

# Operation 01
data = conn.execute("SELECT * FROM product ORDER BY -PRICE")
count = 0
for i in data:
    if count == 3:
        break
    print(i)
    count+=1

# Operation 02
data = conn.execute("SELECT * FROM product WHERE MANUFACTURER = 'ABC' AND
PRICE <= 200")
for i in data:
    print(i)

# Operation 03
data = conn.execute("SELECT * FROM product WHERE PRICE <= 200 and PRICE >
20")
for i in data:
    print(i)

# Operation 04
data = conn.execute("SELECT * FROM product")
count = 0
for i in data:
    count += 1
print(count)

# Operation 05
data = conn.execute("SELECT CATEG FROM product")
category = ""
for i in data:
    if(i[0] != category):
        print(i[0])
        category = i[0]
```