**18CSC207J – Advanced Programming Practice**

# Answer Key

**B2-Set B**

**CT2**

**11.**

```
# 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 = " ")
```

**12.**

Pack Method:

```
from tkinter import *
master = Tk()
# Use of Pack Method
Label(master, text="I'm here to demonstrate Pack Method").pack()
Label(master, text="I'm here to demonstrate Pack Method").pack()
```

Grid Method:

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

# Use of Grid Method
Label(master, text="I'm the first column's label").grid(row = 0)
Label(master, text="I'm the second column's label ").grid(row = 0, column =
1)
```

Place Method:

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

# Use of Place Method
Label(master, text="I'm at the coordinate (x, y) = (0, 0)").place(x = 0, y
= 0)
Label(master, text ="I'm at the coordinates (x, y) = (0, 25)").place(x = 0,
y = 25)
master.mainloop()
```

**13.**

A race condition occurs in a parallel program execution when two or more threads access a common resource, e.g., a variable in shared memory, and the order of the accesses depends on the timing, i.e., the progress of individual threads. The disposition for a race condition is in the parallel program.

Race condition can be prevented using the synchronization in parallel/concurrent programming.
Example:
By using Lock, RLock, Semaphore, etc.
An example of prevention of race condition using Lock:

```
import threading
x = 0      # A shared value
COUNT = 100
lock = threading.Lock()
def incr():
    global x
    lock.acquire()
    print("thread locked for increment cur x=",x)
    for i in range(COUNT):
        x += 1
        print(x)
    lock.release()
    print("thread release from increment cur x=",x)

def decr():
    global x
    lock.acquire()
    print("thread locked for decrement cur x=",x)
    for i in range(COUNT):
        x -= 1
        print(x)
    lock.release()
    print("thread release from decrement cur x=",x)
t1 = threading.Thread(target=incr)
t2 = threading.Thread(target=decr)
t1.start()
t2.start()
t1.join()
t2.join()
```

**14.**

```
# Type 01
listVal = [[1,2,3], [4,5,6], [7,8,9]]
print(*map(lambda x: x, listVal))
```

**Or**

```
# Type 02
listVal = [[1,2,3], [4,5,6], [7,8,9]]
print(*filter(lambda x: x, listVal))
```

**15.**

Map function:

```
topics = ['TopicA', 'TopicB']

tweets = ['This tweet contains TopicA and TopicB', 'This tweet contains
TopicA only', 'This tweet also contains TopicA and TopicB', 'This one too
has TopicA and TopicB']

def findNumberOfTweetsWithBothTopics(a):
    if topics[0] and topics[1] in a:
        return True

# Use of map function
x = 0
val = map(findNumberOfTweetsWithBothTopics, tweets)
print(len([x for i in tuple(val) if i is True]))
```

Filter Function:

```
topics = ['TopicA', 'TopicB']

tweets = ['This tweet contains TopicA and TopicB', 'This tweet contains
TopicA only', 'This tweet also contains TopicA and TopicB', 'This one too
has TopicA and TopicB']

def findNumberOfTweetsWithBothTopics(a):
    if topics[0] and topics[1] in a:
        return True

# Filter Function
val = filter(findNumberOfTweetsWithBothTopics, tweets)
print(len(tuple(val)))
```

Reduce Function:

```
from functools import reduce
topics = ['TopicA', 'TopicB']
tweets = ['This tweet contains TopicA and TopicB', 'This tweet contains
TopicA only', 'This tweet also contains TopicA and TopicB', 'This one too
has TopicA and TopicB']

count = 0
def findNumberofTweets(a, b):
    global count
    if count == 0:
        if topics[0] and topics[1] in a and b:
            count += 1
            return
    if topics[0] and topics[1] in b:
        count += 1

# Reduce Function
reduce(findNumberofTweets, tweets)
print(count)
```

**16.**

```
import sqlite3
conn = sqlite3.connect('testdb.db')
conn.execute('''CREATE TABLE Employee(
                EMP_Id INT PRIMARY KEY NOT NULL,
                NAME TEXT NOT NULL,
                AGE INT NOT NULL,
                SALARY REAL NOT NULL,
                ADDRESS TEXT NOT NULL,
                EMAIL TEXT NOT NULL);''')

records = [
    (101, 'ABC', 22, 10000, 'ZXY', 'abc@mail.com'),
    (102, 'DEF', 24, 20000, 'ZXY', 'abc@mail.com'),
    (103, 'GHI', 21, 30000, 'ZXY', 'abc@mail.com'),
    (104, 'JKL', 26, 45000, 'ZXY', 'abc@mail.com'),
    (105, 'MNO', 30, 21000, 'ZXY', 'abc@mail.com')
]

# operation 01
conn.executemany("INSERT INTO Employee VALUES (?, ?, ?, ?, ?, ?)", records)
print("Datas are successfully inserted")

conn.commit()

# Operation 02
data = conn.execute("SELECT * FROM Employee")
for i in data:
    print(i)

# Operation 03
conn.execute("DELETE FROM Employee WHERE EMP_Id = 101")
data = conn.execute("SELECT * FROM Employee")
for i in data:
    print(i)

# Operation 04
conn.execute("UPDATE Employee SET SALARY = 40000 WHERE EMP_Id = 102")
conn.commit()
data = conn.execute("SELECT * FROM Employee")
for i in data:
    print(i)

# Operation 05
data = conn.execute("SELECT SALARY FROM Employee")
totalSalary = 0
for i in data:
    totalSalary += i[0]
print(totalSalary)
```

**17.**

**a.**

```
from tkinter import *
master = Tk()
master.title("tk")
for i in range(0, 5):
    for j in range(0, 3):
        Button(master, text=f"Row {i} Column {j}").grid(row = i, column =
j)

master.mainloop()
```

**b.**

```
from tkinter import *
master = Tk()
master.title("tk")

Label(master, text = "Username: ").grid(row=0, column = 0)
Entry(master).grid(row = 0, column = 1)

Label(master, text = "Email: ").grid(row=1, column = 0)
Entry(master).grid(row = 1, column = 1)

Label(master, text = "Password: ").grid(row=2, column = 0)
Entry(master).grid(row = 2, column = 1)

master.mainloop()
```

**18.**

```
from multiprocessing import Process

def process1(num):
    print("\nI'm Process 1")
    factorial = 1

    # check if the number is negative, positive or zero
    if num < 0:
        print("Sorry, factorial does not exist for negative numbers")
    elif num == 0:
        print("The factorial of 0 is 1")
    else:
        for i in range(1, num + 1):
            factorial = factorial * i
        print("The factorial of", num, "is", factorial)

def process2(listVal):
    print("\nI'm Process 2")
    print("Even Numbers are: ", end = " ")
    for i in listVal:
        if i%2 == 0:
            print(i, end=" ")
    print("\nOdd Numbers are: ", end=" ")
    for i in listVal:
        if i%2 != 0:
            print(i, end=" ")

def process3(listVal):
```

```
    print("\nI'm Process 3")
    print(reduce(lambda x, y: x * y, listVal))

if __name__ == '__main__':
    p1 = Process(target=process1, args=(3,))
    p2 = Process(target=process2, args=([2,3,4,5,6], ))
    p3 = Process(target=process3, args=([2, 3, 4, 5, 6],))
    p1.start()
    p2.start()
    p3.start()
    p1.join()
    p2.join()
    p3.join()
```

**19.**

```
from functools import reduce
listVal = [2,3,5,6,7,9]
print(reduce(lambda x, y: x + y, listVal))
print(reduce(lambda x, y: x if x > y else y, listVal))

# Part A
stringVal = "AbcD1"
def check_string(str1):
    messg = [
    lambda str1: any(x.isupper() for x in str1) or 'String must have 1
upper case character.',
    lambda str1: any(x.islower() for x in str1) or 'String must have 1
lower case character.',
    lambda str1: any(x.isdigit() for x in str1) or 'String must have 1
number.',
    lambda str1: len(str1) >= 7                 or 'String length should be
atleast 8.',]
    result = [x for x in [i(str1) for i in messg] if x != True]
    if not result:
        result.append('Valid string.')
    return result
stringVal = "AbcD1abd"
print(check_string(stringVal))

# Part B
val = "ABCDEFGHI"
vowels = ['A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o', 'u']
print(*filter(lambda x: x in vowels, val))
```