RAMAIAH
Institute of Technology

Department of Computer Science & Engineering

QUESTION BANK FOR IV SEMESTER (Term: Jan-May 2019)

UNIX and Python Scripting (CS46-1)

I.A. Marks: 50 **Exam Hours: 03**
Credits: 0:1:2:0 **Exam Marks: 50**

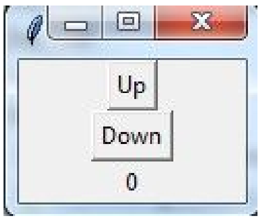
Design the following programs using Python.

1. **(a)** Write a python program to create a file and write contents into the file. Open the file created and count the number of words in the file.

Ans:

```
f=open('example.txt','w')
print('File Opened in write mode:\nWrite the text lines in it and to end press -1')
t=input()
while t!='-1':
    f.write(t+'\n')
    t=input()
f.close()
print('File opened in read mode:')
f=open('example.txt','r')
l=f.readlines()
f.close()
print('The number of lines in the file is',len(l))
```

(b) Write a python program to design a GUI application as shown below. On click of an Up button the value should increment by one and on click of down button the value should decrement by one.



Ans:

```
from tkinter import *
def up():
    v.set(v.get()+1)
def down():
    v.set(v.get()-1)
if __name__=='__main__':
    root=Tk()
    v=IntVar()
    v.set(0)
f=Frame(root).pack()
b1=Button(f,text='UP',command=up).pack()
b2=Button(f,text='DOWN',command=down).pack()
l=Label(f,textvariable=v).pack(side=BOTTOM)
root.mainloop()
```

HOD, Dept. of CSE

2.

(a) Write a python program to create a class called Point , which represents a point. Overload the + operator to add two points. Write functions to read and display the points.

Ans:

```python
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __add__(self, p):
        r = Point(self.x + p.x, self.y + p.y)
        return r
    def __str__(self):
        return '('+str(self.x)+','+str(self.y)+')'
a=Point(1,2)
b=Point(2,3)
c=a+b
print('Point a:',a,'\nPoint b:',b,'\nPoint c=a+b:',c)
```

(b) Write a python function geometric () that takes a list of integers as input and returns True if the integers in the list form a geometric sequence. A sequence a0, a1, a2, a3, a4, . . . , an is a geometric sequence if the ratios a1/a0, a2/a1, a3/a2, a4/a3, . . . , an-1/an are all equal. >>> geometric([2, 4, 8, 16, 32, 64, 128, 256])
True
>>> geometric([2, 4, 6, 8])
False

Ans:

```python
def geometric(l):
    f=0
    r=l[1]/l[0]
    for i in range(2,len(l)):
        if (l[i]/l[i-1])!=r:
            f=1
            break
    if f:
        return False
    return True

print('Enter the list of numbers:')
p=[int(x) for x in input().split()]
print(geometric(p))
```

HOD, Dept. of CSE

3. **(a)** Write a python program to implement the following.

```
Vehicle
  __|___
|        |
Bike    Car
 ___|___
|         |

Pedal bikes  Motor bikes
```

Ans:

```python
class Vehicle:
    def __init__(self,cn):
        self.cn=cn

class Bike(Vehicle):
    def __init__(self,cn,n):
        Vehicle.__init__(self,cn)
        self.n=n

class Car(Vehicle):
    def __init__(self,cn,n,m):
        Vehicle.__init__(self,cn)
        self.n=n
        self.m = m
    def __str__(self):
        return ('Company Name: '+str(self.cn)+'\nCar name: '+str(self.n)+'\nMileage:
'+str(self.m)+'kmpl')

class Pedal_bike(Bike):
    def __init__(self,cn,n,m):
        Bike.__init__(self,cn,n)
        self.m=m
    def __str__(self):
        return ('Company Name: '+str(self.cn)+'\nPedal Bike name:
'+str(self.n)+'\nMileage: '+str(self.m)+'kmpl')

class Motor_bike(Bike):
    def __init__(self,cn,n,m):
        Bike.__init__(self,cn,n)
        self.m=m
    def __str__(self):
        return ('Company Name: '+str(self.cn)+'\nMotor Bike name:
'+str(self.n)+'\nMileage: '+str(self.m)+'kmpl')

c=Car('BMW','i8',47)
p=Pedal_bike('P','p',100)
m=Motor_bike('M','m',90)
print('CAR:',c,'\nPEDAL BIKE',p,'\nMOTOR BIKE',m,sep='\n')
```

HOD, Dept. of CSE

(b) Write a program that creates a Canvas and a Button. When the user presses the Button, it should draw a circle on the canvas.

Ans:

```
from tkinter import *
def draw():
    c.create_oval(50,50,250,250,fill='green')
root=Tk()
c=Canvas(root,height=300,width=300)
b=Button(root,text="Draw Circle",command=draw).pack(side=BOTTOM)
c.pack()
root.mainloop()
```

4. **(a)** Write a python program to demonstrate 'Key Error', 'Value Error', 'Index Error'.

Ans:

```
try:
    l={}
    print('Accessing l[1] from l={}')
    q=l[1]
except KeyError:
    print('Key Error!')
try:
    a=1
    print('\na=1 and changing value to a=int("dog")')
    a=int('dog')
except ValueError:
    print('Value Error!')
try:
    p = [1, 2]
    print('\nAccessing p[3] from p=[1,2]')
    q=p[3]
except IndexError:
    print('Index Error!')
```

(b) Write a python class called Mylist that shadows a python list: it should overload + operator to append the data to the list. Also provide constructor for your class that takes an existing list.

Ans:

```
class Mylist:
    def __init__(self,l=[]):
        self.l=l
    def __add__(self,a):
        self.l.append(a)
        return self.l
    def __str__(self):
        return self.l
l=Mylist([1,2])
print('Adding 3 to the list [1, 2]')
l+=3
print('Updated list:',l)
```

5. **(a)** Write a python program to simulate saving account processing in a bank using constructors. Create Deposit and Withdraw with other member functions and Check for Validation while withdrawing the amount and depositing the amount by defining appropriate user defined exceptions.

Ans:

```
class WithdrawalError(Exception):
    pass
class DepositError(Exception):
    pass
class SavingsAccount:
    def __init__(self,b=500):
        try:
            if b<500:
                raise DepositError
            else:
                self.bal=b
                print('Account created with balance:',self.bal,'\n')
        except DepositError:
            print('Minimum amount to be deposited to create an account is 500.Please deposit Rs.500.\n')
            del(self)
    def deposit(self,a):
        self.bal+=a
        print('Rs.',a,'has been deposited. Your available balance:',self.bal,'\n')
    def withdraw(self,x):
        try:
            if x>self.bal:
                raise WithdrawalError
            else:
                self.bal-=x
                print('Rs.', x, 'has been withdrawn. Your available balance:', self.bal, '\n')
        except WithdrawalError:
            print('You cannot withdraw more than the account balance. Available balance:',self.bal)
u=SavingsAccount(400)
u=SavingsAccount(1000)
u.deposit(10000)
u.withdraw(200000)
u.withdraw(2000)
```

(b) Write a python program to implement Data structure's Stack and Queue using lists by making use of packages. (Note: Create a separate package for stack and Queue).

Ans:

1. mkdir stack (create a directory called stack)
2. cd stack
3. vi __init__.py (save the empty file)
4. vi create.py

```
class Stack:
    def __init__(self,l=[]):
        self.l=l
    def insert(self,x):
        self.l.append(x)
    def top(self):
        return self.l[-1]
    def remove(self):
        self.l.pop()
    def __str__(self):
        return str(self.l)
```

5. cd .. (come out of stack directory)
6. mkdir queue (create a directory called queue)
7. cd queue
8. vi __init__.py (save the empty file)
9. vi create.py

```
 class Queue:
    def __init__(self,l=[]):
        self.l=l
    def insert(self,x):
        self.l.append(x)
    def front(self):
        return self.l[0]
    def rear(self):
        return self.l[-1]
    def remove(self):
        self.l.pop(0)
    def __str__(self):
        return str(self.l)
```

10. cd ..
11. vi program.py

```
from stack.create import Stack
from queue.createq import Queue
a=Stack([1,2,3,4])
b=Queue([1,2,3,4])
print('Stack:',a)
print('Top:',a.top())
print('Inserting 5 to a')
a.insert(5)
print(a)
print('Popping out top')
a.remove()
print(a)
print('\n\n')
print('Queue:',b)
print('Front:',b.front())
print('Rear:',b.rear())
print('Inserting 5 to b')
b.insert(5)
print(b)
print('Popping out front')
b.remove()
print(b)
print('\n\n')
```

6. **(a)** Write a python program to create a database Employee with relevant data and display the Employees whose salary > 50000.

Ans:

```
from sqlite3 import *
open('Employee.db','w').close()
con=connect('Employee.db')
cur=con.cursor()
cur.execute('CREATE TABLE IF NOT EXISTS Employee (Name TEXT,Salary INTEGER)')
cur.execute('INSERT INTO Employee VALUES("A",20000)')
cur.execute('INSERT INTO Employee VALUES("B",60000)')
cur.execute('INSERT INTO Employee VALUES("C",90000)')
cur.execute('INSERT INTO Employee VALUES("D",70000)')
cur.execute('INSERT INTO Employee VALUES("E",10000)')
con.commit()
cur.execute('SELECT* FROM Employee WHERE Salary>50000')
l=cur.fetchall()
print('NAME\tSALARY')
for i in l:
    print(i[0]+'\t\t'+str(i[1]))
```

(b) Write a python function exclamation() that takes input as a string and returns it with this modification: Every vowel is replaced by four consecutive copies of itself and an exclamation mark (!) is added at the end.

```
>>> exclamation('argh')
'aaaargh!'
>>> exclamation('hello') 'heeeelloooo!
```

HOD, Dept. of CSE

Ans:

```
from re import *
s=input('Enter the string: ')
p='[aeiou]'
def exclamation(s):
    m=search(p,s,flags=IGNORECASE)
    if m:
        s = sub('a', 'aaaa', s, IGNORECASE)
        s = sub('e', 'eeee', s, IGNORECASE)
        s = sub('i', 'iiii', s, IGNORECASE)
        s = sub('o', 'oooo', s, IGNORECASE)
        s = sub('u', 'uuuu', s, IGNORECASE)
    s+='!'
    return s
s=exclamation(s)
print(s)
```

7. **(a)** Create a python dictionary for words and their meanings. Write functions to add a new entry (word:meaning) ,search for a particular word and retrieve meaning, given meaning find words with same meaning , remove an entry, display all words sorted alphabetically. [Program must be menu driven]

Ans:

```
d={}
def new(w,m):
    d[w]=m
def retrieve(w):
    if w not in d:
        return -1
    return d[w]
def same_meaning(m):
    l=[]
    for i,j in d.items():
        if j==m:
            l.append(i)
    if len(l)==0:
        return -1
    return l
def remove(w):
    if w not in d:
        return -1
    d.pop(w)
    return 1
def display_sorted():
    print("DICTIONARY:")
    for i in sorted(d.keys()):
        print(i,':',d[i])
while(1):
    print()
    print('DICTIONARY MENU'.center(25,'*'))
    c=int(input('1. New Entry\n2. Retrieve meaning for a word\n3. Words with same
meaning\n4. Remove an entry\n5. Display the dictionary\n6. Exit\nEnter your choice: '))
    if c==1:
        w,m=input('Enter word and its meaning (eg., word : meaning): ').split(' : ')
        new(w,m)
        print('New entry added!')
    elif c==2:
        w=input('Enter the word: ')
        k=retrieve(w)
        if k==-1:
            print('No such word in the dictionary')
        else:
            print('Meaning:',k)
    elif c==3:
        m=input('Enter the meaning: ')
        k=same_meaning(m)
        if k==-1:
            print('No words in the dictionary mean that!')
        else:
            for i in k:
                print(i,end=' ')
            print('(is/are) the word(s) that mean it')
    elif c==4:
        w=input('Enter the word to be removed: ')
        k=remove(w)
        if k==-1:
            print('No such word in the dictionary to be removed!')
        else:
            print(w,'is removed from the dictionary')
    elif c==5:
        display_sorted()
    elif c==6:
        break
```

(b) Write a python function subsetSum() that takes as input a list of positive numbers and a positive number target. Your function should return True if there are three numbers in the list that add up to target. For example, if the input list is [5, 4, 10, 20, 15, 19] and target is 38, then True should be returned since 4+15+19 = 38. However, if the input list is the same but the target value is 10, then the returned value should be False because 10 is not the sum of any three numbers in the given list.

Ans:

```
def remove(l,r): #for unique elements of x, y and z in list
    k=[]
    k+=l
    if r in l:
        k.remove(r)
    return k
def subsetsum(l,tar):
    k=[(x,y,z) for x in l for y in remove(l,x) for z in remove(remove(l,x),y) if
x+y+z==tar]
    if len(k):
        return True
    return False
print('Enter the list elements: ',end='')
lst=[int(x) for x in input().split()]
t=int(input('Enter the target: '))
print(subsetsum(lst,t))
```

8. **(a)** Define a python function generate_n_chars() that takes an integer n and a character c and returns a string, n characters long. For example, generate_n_chars(5,"x") should return the string “xxxxx” using keyword only parameters.

Ans:

```
def generate_n_chars(n,s):
    return s*n
st=input('Enter a character: ')
i=int(input('Enter n: '))
print(generate_n_chars(i,st))
```

(b) Create a python module named Area which has functions to compute area of circle, rectangle and scalene triangle. Function to compute area of circle takes radius as argument, function to compute area of rectangle takes length and breadth as arguments and function to compute area of triangle takes lengths of three sides as arguments. Write client code (menu driven) which imports module Area, reads the input from user and invoke appropriate functions.

Ans:

1. vi Area.py

```
from math import *
def area_circle(r):
    return 4*atan(1)*(r**2)
def area_rectangle(l,b):
    return l*b
def area_triangle(a,b,c):
    s=(a+b+c)/2
    return sqrt(s*(s-a)*(s-b)*(s-c))
```

2. vi client.py

```
from Area import *
while(1):
    print()
    print('Area Menu'.center(19,'*'))
    c=int(input('1. Area of circle\n2. Area of rectangle\n3. Area of a triangle\n4. Exit\nEnter your choice: '))
    if c==1:
        r=float(input('Enter the radius of the circle: '))
        print('Area:',area_circle(r))
    elif c==2:
        print('Enter length and breadth of the rectangle: ',end='')
        s=[float(x) for x in input().split()]
        l=s[0]
        b=s[1]
        print('Area:',area_rectangle(l,b))
    elif c==3:
        print('Enter the sides of the triangle ', end='')
        s = [float(x) for x in input().split()]
        a = s[0]
        b = s[1]
        c = s[2]
        print('Area:', area_triangle(a,b,c))
    elif c==4:
        break
```

HOD, Dept. of CSE

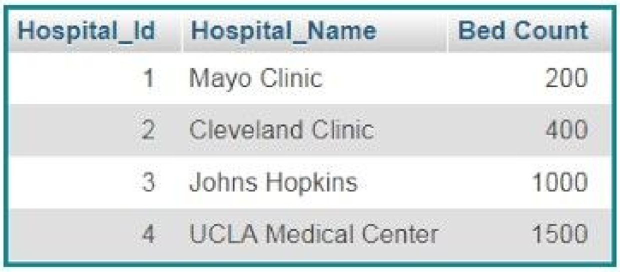
9.

(a) Write a python program to create a class called time. Its three members all type int should be called hours, minutes and seconds. Your program should prompt the user to enter a time values separately. The Program should then store the time in the object and finally printout the total no of seconds represented by this value. Use appropriate member functions.

Ans:

```
class time:
    def __init__(self,h=0,m=0,s=0):
        self.hours=h
        self.minutes=m
        self.seconds=s
    def set_time(self,h,m,s):
        self.hours = h
        self.minutes = m
        self.seconds = s
    def __str__(self):
        return 'Time: '+str(self.hours)+':'+str(self.minutes)+':'+str(self.seconds)
    def get_seconds(self):
        return self.hours*3600+self.minutes*60+self.seconds
t=time()
h,m,s=input('Enter hours, minutes and seconds: ').split()
h=int(h)
m=int(m)
s=int(s)
t.set_time(h,m,s)
print(t,'has',t.get_seconds(),'seconds')
```

(b) Write a python program to create a database Hospital with the table and attributes as shown below. Execute queries for updating the Bed count of Hospital_Id 1 to 100 and delete the row with the Hospital_Id 3.



Hospital_Id	Hospital_Name	Bed Count
1	Mayo Clinic	200
2	Cleveland Clinic	400
3	Johns Hopkins	1000
4	UCLA Medical Center	1500

Ans:

```
from sqlite3 import *
open('Hospital.db','w').close()
con=connect('Hospital.db')
cur=con.cursor()
cur.execute('CREATE TABLE IF NOT EXISTS Hospital (Hospital_Id int,Hospital_Name
text,Bed_Count int)')
cur.execute('INSERT INTO Hospital VALUES(1,"Mayo Clinic",200)')
cur.execute('INSERT INTO Hospital VALUES(2,"Cleveland Clinic",400)')
cur.execute('INSERT INTO Hospital VALUES(3,"Johns Hopkins",1000)')
cur.execute('INSERT INTO Hospital VALUES(4,"UCLA Medical Centre",1500)')
con.commit()
print('HOSPITAL DB'.center(31,'*'))
print('ID\t\tHospital Name\t\tBed Count')
cur.execute('SELECT* FROM Hospital')
l=cur.fetchall()
for i in l:
    print(i[0],'\t',i[1].center(19),'\t\t',i[2],sep='') #19 because the length of
largest hospital name is 19

print('\nExecuting suggested queries:')
print('1. Updating Bed Count of Hospital ID 1 to 100')
```

```python
cur.execute('UPDATE Hospital SET Bed_Count=100 WHERE Hospital_Id=1')
con.commit()
print('HOSPITAL DB'.center(31,'*'))
print('ID\t\tHospital Name\t\tBed Count')
cur.execute('SELECT* FROM Hospital')
l=cur.fetchall()
for i in l:
    print(i[0],'\t',i[1].center(19),'\t\t',i[2],sep='') #19 because the length of
largest hospital name is 19

print('\n2. Deleting entry of Hospital ID 3')
cur.execute('DELETE FROM Hospital WHERE Hospital_Id=3')
con.commit()
print('HOSPITAL DB'.center(31,'*'))
print('ID\t\tHospital Name\t\tBed Count')
cur.execute('SELECT* FROM Hospital')
l=cur.fetchall()
for i in l:
    print(i[0],'\t',i[1].center(19),'\t\t',i[2],sep='') #19 because the length of
largest hospital name is 19
```

10. **(a)** Write a python class to represent city which contains a list of places to see. Provide methods to create the object with just the city name or with city name and places (stored as list) Provide methods to add a place of visit, to remove place of visit, to display all places of visit.

Ans:

```python
class city:
    def __init__(self,name,places=[]):
        self.cn=name
        self.pl=places
    def add_place(self,l):
        self.pl.append((l))
    def remove_place(self,l):
        self.pl.remove(l)
    def places(self):
        print("Places to visit in",self.cn,": ",end='')
        for i in self.pl:
            print(i,'\t\t',sep='',end='')
        print()
b=city('Bengaluru',['Yeshwantpur','Lalbagh','Cubbon Park'])
print('A city "b" is initialised named "Bengaluru":')
b.places()
print('\nAdding a place to visit "Wonderla" into the city "b":')
b.add_place('Wonderla')
b.places()
print('\nRemoving a place to visit "Yeshwantpur" from the city "b":')
b.remove_place('Yeshwantpur')
b.places()
```

(b) Write a python function exclamation() that takes input as a string and returns it with this modification: Every vowel is replaced by four consecutive copies of itself and an exclamation mark (!) is added at the end.

```
>>> exclamation('argh')
'aaaargh!'
>>> exclamation('hello') 'heeeelloooo!
```

Ans:

Same as 6(b)

HOD, Dept. of CSE

11. **(a)** Write a python function names() that takes no input and repeatedly asks the user to enter the first name of a student in a class. When the user enters the empty string, the function should print for every name the number of students with that name.

>>> names()
Enter next name: Valerie
Enter next name: Bob
Enter next name: Valerie
Enter next name: Amelia
Enter next name: Bob Enter
next name:
There is 1 student named Amelia
There are 2 students named Bob
There are 2 students named Valerie

Ans:

```
from collections import defaultdict
d=defaultdict(int)
def names():
    while(1):
        n=input('Enter the first name: ')
        if n=='':
            break
        d[n]+=1
    for i in sorted(d.keys()):
        if d[i]<2:
            print('There is 1 student named',i)
        else:
            print('There are',d[i],'students named',i)
names()
```

(b) Write a python function subsetSum() that takes as input a list of positive numbers and a positive number target. Your function should return True if there are three numbers in the list that add up to target. For example, if the input list is [5, 4, 10, 20, 15, 19] and target is 38, then True should be returned since 4+15+19 = 38. However, if the input list is the same but the target value is 10, then the returned value should be False because 10 is not the sum of any three numbers in the given list.

Ans:

Same as 7(b)

12. **(a)** Write a python function partition() that splits a list of soccer players into two groups. More precisely, it takes a list of first names (strings) as input and prints the names of those soccer players whose first name starts with a letter between and including A and M. >>> partition(['Eleanor', 'Evelyn', 'Sammy', 'Owen', 'Gavin']) Eleanor
Evelyn
Gavin
>>> partition(['Xena', 'Sammy', 'Owen'])

Ans:

```
def partiton(l):
    for i in l:
        if i.upper()>='A' and i.upper()<='M':
            print(i)
print('Enter the first names of soccer players: ',end='')
l=[x for x in input().split()]
partiton(l)
```

(b) Write a python program that creates a GUI with a single button. When the button is pressed it should create a second button. When that button is pressed, it should create a label that says, “Nice job!..

Ans:

```
from tkinter import *
def second():
    l=Label(root,text='Nice Job!',fg='green').pack()
def first():
    b2=Button(root,text='Second',command=second).pack()
root=Tk()
b1=Button(root,text='First',command=first).pack()
root.mainloop()
```

13. **(a)** Write a python program for , A string with parentheses is well bracketed if all parentheses are matched: every opening bracket has a matching closing bracket and vice versa. Write a Python function wellbracketed(s) that takes a string s containing parentheses and returns True if s is well bracketed and False otherwise. Here are some examples to show how your function should work. >>> wellbracketed("22)")

False

>>> wellbracketed("(a+b)(a-b)")

True

>>> wellbracketed("(a(b+c)-d)((e+f)")

False

Ans:

```
from re import *
def wellbracketed(s):
    l=split('\w+',s)
    lp=0
    rp=0
    for i in l:
        if i=='(':
            lp+=1
        elif i==')':
            rp+=1
    if lp==rp:
        return True
    return False
s=input('Enter the string: ')
print(wellbracketed(s))
```

(b) Write a python program to demonstrate, a list rotation consists of taking the last element and moving it to the front. For instance, if we rotate the list [1,2,3,4,5], we get [5,1,2,3,4]. If we rotate it again, we get [4,5,1,2,3].Write a Python function rotatelist(ls,k) that takes a list ls and a positive integer k and returns the list ls after k rotations. If k is not positive, your function should return ls unchanged. Note that your function should not change ls itself, and should return the rotated list.Here are some examples to show how your function should work.

>>> rotatelist([1,2,3,4,5],1) #output is [5, 1, 2, 3, 4]
>>> rotatelist([1,2,3,4,5],3) #output is [3, 4, 5, 1, 2]
>>> rotatelist([1,2,3,4,5],12) #output is [4, 5, 1, 2, 3]

HOD, Dept. of CSE

Ans:

```
def rotatelist(l,n):
    if n==0:
        return l
    k=[l[-1]]
    for i in range(len(l)-1):
        k.append(l[i])
    return rotatelist(k,n-1)
print('Enter the list of elements: ',end='')
lst=[int(x) for x in input().split()]
i=int(input('Enter the number of rotations to be done: '))
print(rotatelist(lst,i))
```

14. **(a)** Write a python program to validate name and phone number using re. It will continue to ask until you put correct data only. (eg.Phone number: (800) 555.1212 #1234.

Ans:

```
#eg.Phone number: (800) 555.1212 #1234
from re import *
while(1):
    n,s=input('Enter a name and phone-number (eg, abcd : phonenumber): ').split(' : ')
    r=search(r'\(\d{3}\)\ \d{3}\.\d{4}\ \#\d{4}',s)
    if r:
        print('Validated correctly!')
        break
    else:
        print('Please enter a valid phone-number')
```

(b) Write a python program Convert the contents of the file 'graffit.txt' to all uppercase letters.

Ans:

1. **Create a text file 'graffit.txt'**
2. **save some text lines with all lower cases**
3. **program.py**

```
f=open('graffit.txt','r')
l=[]
for i in f.readlines():
    l.append(i.upper())
f.close()
f=open('graffit.txt','w')
for i in l:
    f.write(i)
f.close()
```

15. **(a)** Write a "spelling correction" python function correct() that takes a string and sees to it that two or more occurrences of the space character is compressed into one, and inserts an extra space after a period if the period is directly followed by a letter. (use regular expression) E.g. correct("This   is   very funny   and   cool.Indeed!") should return "This is very funny and cool. Indeed!"

Ans:

```
from re import *
def correct(s):
    s=sub(r'\ +',r' ',s)
    s=sub(r'\.',r'. ',s)
    return s
s=input('Enter the string: ')
print(correct(s))
```

(b) Write a python program to create list1 and list2, be two lists of integers. We say that list1 is a sub-list of list2 if the elements in list1 appear in list2 in the same order as they appear in list1, but not necessarily consecutively.

>>> sublist([15, 1, 100], [20, 15, 30, 50, 1, 100])
True
>>> sublist([15, 50, 20], [20, 15, 30, 50, 1, 100]) False

Ans:

```
def sublist(a,b):
    ind=0
    for i in a:
        if i not in b[ind:]:
            return False
        else:
            ind=b.index(i)
    return True
print('Enter list 1 elements: ',end='')
l1=[int(x) for x in input().split()]
print('Enter list 2 elements: ',end='')
l2=[int(x) for x in input().split()]
print(sublist(l1,l2))
```

Note:

UNIX quiz	**: 15 Marks**
Write up	**: 08 Marks**
Conduction and Result	**: 20 Marks (a: 10 Marks, b: 10 Marks)**
Viva	**: 07 Marks**
For Change of question	**: -10 Marks**

HOD, Dept. of CSE