Computer Science (083)

Python
&
SQL

Practical Record - Grade XII (2017-2018)

Name : _____

Class : _____ Section : _____

Reg No. : _____

Name of Teacher : _____

جيمس مدرستنا الهندية
GEMS OUR OWN INDIAN SCHOOL

COMPUTER SCIENCE JOURNAL
GRADE XII – CBSE

REGISTER NO: ……

This is certified to be the Bonafide Computer Science journal of the work done by Miss / Master …….. in the school Laboratory during the year 2017-2018.

Principal **Teacher- in- charge**

Submitted for the practical examination on __ /__ / 2018 held at GEMS Our Own Indian School, Dubai.

Examiners

1) ……..……

2) ……..……..

Dubai
2017-2018

Part-I
Python Programming

Part-II
SQL Programming

PART I

Python

Programming Language

Program No: 1

A Program to play Rock Paper Scissor Game.

Program Code

```
import random
def number_to_name(number):
    if number == 0:
        name = "Rock"
    elif number == 1:
        name = "paper"
    elif number == 2:
        name = "Scissors"
    return name
def name_to_number(name):
    if name == "Rock":
        number = 0
    elif name == "Paper":
        number = 1
    elif name == "Scissors":
        number = 2
    else:
         number=-1
    return number
def rpsls(name):
    comp_number = random.randrange(0,3)
    player_number = name_to_number(name)
    comp_move = number_to_name(comp_number)
    print "The Computer chose:",comp_move
    print "Player chose:", name
```

```
    if (comp_number - player_number) % 3 == 0:
      print "Computer and Player tie!"
    elif ( comp_number-player_number ) % 3 == 1:
        print "Computer wins!"
        print
    else:
        print "Player Wins"
# function calling
rpsls("Rock")
rpsls("Paper")
rpsls("Scissors")
```

Output

```
The Computer chose: Scissors
Player chose: Rock
Player Wins

The Computer chose: Scissors
Player chose: Paper
Computer wins!

The Computer chose: Scissors
Player chose: Scissors
Computer and Player tie!
```

Program No: 2

A program to play Guessing Game

Program Code

```
import random
secretnum= random.randint(1,100)  # generates random number
#print(secretnum)
guessnum=int(input("Enter the number between 1 to 100:"))
while guessnum!=secretnum :
    if(guessnum <secretnum) :
            print "your guess is too low "
    else:
            print "your guess is too high "
    guessnum=int(input("Enter the number :"))
print("Congrats: You Found the Number")
```

Output

```
Enter the number between 1 to 100:5
your guess is too low
Enter the number :65
your guess is too high
Enter the number :60
your guess is too low
Enter the number :63
your guess is too high
Enter the number :62
your guess is too high
Enter the number :61
Congrats: You Found the Number
```

Program No: 3

A Program to calculate area of different shapes.

Program Code

```
import math
while (True):
    Area = 0
    print ("1. Area of a circle")
    print ("2. Area of a rectangle")
    print ("3. Area of a square")
    print ("4. Quit")
    Opt = int(input( "Enter your option: "))
    if (Opt == 1) :
        r = float(input("Enter radius of the circle: "))
        Area = round(math.pi, 2) * r * r
        print("Area of the circle is => %.2f" % Area)
    elif (Opt == 2) :
        length = float(input("Enter length of the rectangle: "))
        breadth = float(input("Enter breadth of the rectangle: "))
        Area = length * breadth
        print("Area of the rectangle is => %.2f" % Area)
    elif (Opt == 3) :
        side = float(input("Enter side of the square: "))
        Area = side * side
        print("Area of the square is => %.2f" % Area)
    if (Opt == 4):
        exit()
```

Output

```
1. Area of a circle
2. Area of a rectangle
3. Area of a square
4. Quit
Enter your option: 1
Enter radius of the circle: 24
Area of the circle is => 1808.64

1. Area of a circle
2. Area of a rectangle
3. Area of a square
4. Quit
Enter your option: 2
Enter length of the rectangle: 4
Enter breadth of the rectangle: 6
Area of the rectangle is => 24.00

1. Area of a circle
2. Area of a rectangle
3. Area of a square
4. Quit
Enter your option: 3
Enter side of the square: 5
Area of the square is => 25.00
```

Program No: 4

A program to check the special numbers.

Program Code

```
import math
while (True):
    print ("M A I N M E N U")
    print ("- - - - - - - - - - -")
    print ("1. Palindrome Number")
    print ("2. Armstrong number")
    print ("3. Prime number")
    print ("4. Quit")
    print ("- - - - - - - - - - -")
    Opt = int(input( "Enter your option: "))
    if (Opt == 1) :
        N = int(input("Enter a number to check special number: "))
        N1 = N
        Rem = Rev = 0
        while (N1 > 0):
            Rem = N1 % 10
            Rev=Rev*10+Rem;
            N1 = N1 // 10 #Floor division
        if (N == Rev):
            print ("It is a Palindrome Number.")
        else:
            print ("It is not Palindrome Number.")
        raw_input()
```

```
elif (Opt == 2) :
    N = int(input("Enter a number to check armstrong number: "))
    N1 = N
    Rem = Sum = 0
    while (N1 > 0):
      Rem = N1 % 10
      N1 = N1 // 10 #Floor division
      Sum = Sum + math.pow(Rem, 3)
    if (N == Sum):
        print ("It is an armstrong number.")
    else:
        print ("It is not an armstrong number.")
    raw_input()

  elif (Opt == 3) :
    Opt = 0
    N = int(input("Enter a number to check prime number: "))
    for i in range(2, N):
      if (N % i == 0):
        Opt = 1
        break
    if (Opt == 1):
      print ("It is not a Prime number")
    else:
      print ("It is a prime number.")
    raw_input()
  elif (Opt==4):
    exit()
```

Output

M A I N M E N U

- - - - - - - - - - -

1. Palindrome Number
2. Armstrong number
3. Prime number
4. Quit

- - - - - - - - - - -

Enter your option: 1

Enter a number to check special number: 12321 It is a Palindrome Number.

M A I N M E N U

- - - - - - - - - - -

1. Palindrome Number

2. Armstrong number
3. Prime number
4. Quit

- - - - - - - - - - -

Enter your option: 2

Enter a number to check armstrong number: 153 It is an armstrong number.

Program No: 5

A python program for word jumble game where computer will randomly pick a word from a sequence (tuple) .The user would guess a word from the tuple, if it is correct then print an appropriate message. The tuple is as follows: ("TRANSMISSION","NETWORK","TOPOLOGY","ETHERNET")

Program Code

```
import random
WORDS = ("TRANSMISSION","NETWORK","TOPOLOGY","ETHERNET")
word=random.choice(WORDS)
correct  =word
correct = correct.lower()
jumble = " "
while word:
    position=random.randrange(len(word))
    jumble += word[position]
    word = word[:position] + word[(position+1):]
     print \
     print "The jumble word is:",jumble
Guess = raw_input("Guess the word:")
if Guess==correct :
     print "Your answer is correct“
else:
  print ("Sorry, %s is not exactly jumble " %jumble)
  Guess = raw_input("Please guess again:")
  Guess = Guess.lower()
  if  Guess == correct:
      print "Thats it! You Guessed it! \n"
  else:
        print 'wrong'
```

Output

The jumble word is: TREHEENT

Guess the word:rr

Sorry, TREHEENT is not exactly jumble

Please guess again:ethernet

Thats it! You Guessed it!

Program No: 6

A program to generate students progress report card

Program Code

```
ctr ,Total,Per,Grd=1,0,0,""
Grade=""
lst=[]
t=()
def Cal_Grade(Per):
        Grd=""
        if Per > 91 : Grd = "A1"
        elif Per > 81 and Per <= 90: Grd = "A2"
        elif Per > 71 and Per <= 80: Grd = "B1"
        elif Per > 61 and Per <= 70: Grd = "B2"
        elif Per > 51 and Per <= 60: Grd = "C1"
        elif Per > 41 and Per <= 50: Grd = "C2"
        elif Per > 33 and Per <= 40: Grd = "D"
        return Grd

print ("Student  List....")
print ("-" * 110)
print "Name                ", "Subject 1", " Subject 2", " Subject 3", " Subject 4 ",
"  Subject 5 ","  Total ", "Percentage ", " Grade"
print ("-" * 110)
#               Entering 10 best students information
print ctr
```

```
while ctr <5  :
        print
        Name = raw_input("Enter Name: ")
        Name = Name.upper()
        Sub1 = float(input("Enter first subject marks: "))
        Sub2 = float(input("Enter second subject marks: "))
        Sub3 = float(input("Enter third subject marks: "))
        Sub4 = float(input("Enter fourth subject marks: "))
        Sub5 = float(input("Enter fifth subject marks: "))
        Total=Sub1+Sub2+Sub3+Sub4+Sub5
        Per=Total/5
        Grade=Cal_Grade(Per)
        ctr += 1
        t=(Name,Sub1,Sub2,Sub3,Sub4,Sub5,Total,Per,Grade)
        lst.append(t)

print "Name                ", "Subject 1", " Subject 2", " Subject 3", " Subject 4 ",
"  Subject 5 ","  Total ", "Percentage ", " Grade"

for i in lst:
        Name,Sub1,Sub2,Sub3,Sub4,Sub5,Total,Per,Grade=i

        print Name+"        ", str(Sub1)  +"     " , str(Sub2)+"    ",  str(Sub3)
+"  ", str(Sub4 )+"   ",  str(Sub5 )+ "  " , str(Total )  +"   ", str(Per )+"  " +
str(Grade)

print ("-" * 110)
```

Output

```
Name                Subject 1  Subject 2  Subject 3
Subject 4    Subject 5    Total  Percentage   Grade

KEVIN          78.0      67.0     88.0   56.0    99.0   388.0
77.6  B1

DEVIKA          67.0      99.0     87.0   78.0    56.0
387.0    77.4  B1

KHAIRAH          87.0      66.0     91.0   23.0    44.0
311.0    62.2  B2

ELWY          56.0      88.0     7.0   98.0    67.0   316.0
63.2  B2

------------------------------------------------------------------------
-------------------------------------

>>>
```

Program No: 7

A python program for entering a Dictionary containing Name and Number.

Program Code

```
print '~'*80
n=int(input("Enter total: "))
phonebook={}
i=1
while i<=n:
    name=raw_input("Enter name : ")
    no=raw_input("Enter Numb: ")
    phonebook[name]=no
    i+=1
sname=raw_input("Enter Name to Search: ")
n=1
for x in phonebook.keys():
    if cmp(x,sname)==0:
        print x,'\t-\t',phonebook[x]
        n=0
if n==1:
        print "Name do not Exist"
raw_input()
```

Output

```
Enter total: 3
Enter name : kevin
Enter Numb: 055-3455456
Enter name : devika
Enter Numb: 050-5935944
Enter name : Daddy
Enter Numb: 04-6544432
Enter Name to Search: kevin
kevin  -                055-3455456
```

Program No: 8

A program to generate pay slip for an employee using OOPS concept.

Program Code

```
# Accessing employee information to calculate gross salary
class Employee:
#Constructor called when creating an object of class type
        def __init__(self, ecode, ename, edesig, esal):
                self.empcode = ecode
                self.empname = ename
                self.empdesig = edesig
                self.empbasic = esal
#               DA calculation method
        def Calc_DA(self):
          self.__da = self.empbasic * 0.8 # _da is a private attribute
          return self.__da
        # HRA calculation method
        def __Calc_HRA(self):
          self.__hra = self.empbasic * 0.1 # _hra is a private attribute
          return self.__hra
        def Calc_Gross(self):
        # Accessing attributes and methods in class method
                empda=self.Calc_DA() # Calling public method thorough
        #reference
                emphra = self.__Calc_HRA() # Calling private method
thorough reference
                empgross = self.empbasic + empda + emphra
                return empgross
        # A class method to display employee information
```

```
def Display(self):
        print ("Here is the employee information....")
        print ("=" * 30)
        print("Code:", self.empcode)
        print("Name:", self.empname)
        print("Designation:", self.empdesig)
        print("Basic salary:", self.empbasic)

        print("Dearness Allowance(DA): %.2f" % self.Calc_DA())
        print("House Rent Allowance(HRA): %.2f" % self.__Calc_HRA())
        print("Gross income: %.2f" % self.Calc_Gross())

# Main program starts

# Enter employee data

print("Enter employee data")

ecode = raw_input("Enter code: ")
ename = raw_input("Enter name: ")
edesig = raw_input("Enter designation: ")
esal = float(input("Enter basic salary: "))

Emp = Employee(ecode, ename, edesig, esal)

# Calling the class method using instance object
Emp.Display()
```

Output

```
Enter employee data
Enter code: 120
Enter name: Alvin
Enter designation: Engineer
Enter basic salary: 200000
Here is the employee information....
==============================
('Code:', '120')
('Name:', 'Alvin')
('Designation:', 'Engineer')
('Basic salary:', 200000.0)
Dearness Allowance(DA): 160000.00
House Rent Allowance(HRA): 20000.00
Gross income: 380000.00
```

Program No: 9

A program to convert decimal to binary.

Program Code

```
num=int(input("Enter Value : "))
print
m=''
for x in range (num):
    print '2|',num,'\t|',num%2
    m+=str(num%2)
    num=num//2
    if num<=0:
        break
binary=''
for x in range (len(m)-1,-1,-1):
    binary+=m[x]

print '\nBinary : ' , binary
```

Output

Enter Value : 112

2| 112 | 0

2| 56 | 0

2| 28 | 0

2| 14 | 0

2| 7 | 1

2| 3 | 1

2| 1 | 1

Binary : 1110000

Program No: 10

A program to generate pay slip for an employee(Teacher) using inheritance concept.

Program Code

```
# Person base class or super class
class Person:
    def __init__(self, name, age, psex):
        self.pname = name
        self.page = age
        self.psex = psex
    def Person_inputData(self):
        self.pname = raw_input("Enter name: ")
        self.page = input("Enter age: ")
        self.psex = raw_input("Enter gender: ")
        print()
    def Person_Display(self):
        print ("Person's information....")
        print ("==========================")

        print ("Teacher detail is:")
        print ("==========================")
        print "Name: ", self.pname
        print "Age: ", self.page
        print "Sex: ", self.psex
# Employee base class or super
class  Employee:
    def __init__(self, ecode, edesig,  esalary):
        self.empcode = ecode
        self.empdesig = edesig
        self.empsalary = esalary
```

```
    def Emp_inputData(self):
        self.empcode = raw_input("Enter code: ")
        self.empdesig = raw_input("Enter department: ")
        self.empsalary = raw_input("Enter basic salary: ")
    def Emp_Display(self):
        print "Code: ", self.empcode
        print "Department: ", self.empdesig
        print "Salary: ", self.empsalary
# Derived class for multiple inheritance
class Teacher(Person, Employee):
    def __init__(self, a, b, c, d, e, f):
         Person.__init__(self, a, b, c)
         Employee.__init__(self, d, e, f)
    def get_data(self):
        Person.Person_inputData(self)
        Employee.Emp_inputData(self)

# Invoking base class constructors
# Invoking base class methods
    def show_data(self):

# Invoking base class methods
        Person.Person_Display(self)
        Employee.Emp_Display(self)
# Main program

T = Teacher(' ', 0, ' ', ' ', ' ', 0)

T.get_data()

T.show_data()
raw_input()
```

Output

```
Enter name: John
Enter age: 32
Enter gender: Male
()
ENTER EMPLOYMENT DETAILS:
Enter employee code: 234
Enter department: Dance
Enter basic salary: 20000
Person's information....
==========================
Teacher detail is:
==========================
Name:  John
Age:  12
Sex:  Male
Code:  234
Department:  Dance
Salary:  20000
```

Program No: 11

A program to use the concept of Exception Handling.

Program Code

```
class Error(Exception):
    pass
class ValueTooSmallError(Error):
    pass
class ValueTooLargeError(Error):
    pass
number=int(input("Enter Number: "))
while True:
    try:
        i_num=int(input("Enter a Number: "))
        if i_num<number:
            raise ValueTooSmallError
        elif i_num>number:
            raise ValueTooLargeError
        break
    except ValueTooSmallError:
        print "This Value Too Small, try again !"
    except ValueTooLargeError:
        print "This Value Is Too Large, try again!"
```

Output

```
Enter Number: 124
Enter a Number: 6698
This Value Is Too Large, try again!
Enter a Number: 333333333
This Value Is Too Large, try again!
Enter a Number: 1
This Value Too Small, try again !
Enter a Number: 666
This Value Is Too Large, try again!
Enter a Number:
```

Program No: 12

A program for performing different types of sorting.

Program Code

```
def  bubble_sort(lst,size):


    for i in range(1,size):
        print "\nIteration:",i
        for j in range(0,size-1):
            if lst[j]>lst[j+1]:
                temp=lst[j]
                lst[j]=lst[j+1]
                lst[j+1]=temp
        for k in lst:
            print k,



def select_sort(lst,size):
    for i in range(size-1):
        print "\nIteration:",i+1
        small=lst[i]
        for j in range(i+1,size):
            if small>lst[j]:
                pos=j
                small=lst[j]
                temp=lst[pos]
                lst[pos]=lst[i]
                lst[i]=temp
        for k in lst:
            print k,
```

```
def  insert_sort(lst,size):
    for i in range(1,size):
        print "\nIteration:",i
        temp=lst[i]
        pos=i-1
        while pos>=0 and lst[pos]>temp:
            lst[pos+1]=lst[pos]
            pos-=1
            lst[pos+1]=temp
        for k in lst:
            print k,
def  main():
    while True:
        print
        print("Main Menu: Sorting")
        print("------------------")
        print("1 - Bubble Sorting")
        print("2 - Selection Sorting")
        print("3 - Insertion Sorting")
        print("4 - Quit")
        print("------------------")
        opt=input("Enter you choice: ")
        if opt==1:
            lst=[]
            size=input("Enter the size of list-1:")
            for i in range(size):
                ele=input("Enter the element :")
                lst.append(ele)
            bubble_sort(lst,size)
```

```
        elif opt==2:
            lst=[]
            size=input("enter the size of list-1:")
            for i in range(size):
                ele=input("Enter the element :")
                lst.append(ele)
            select_sort(lst,size)
        elif opt==3:
            lst=[]
            size=input("enter the size of list-1:")
            for i in range(size):
                ele=input("Enter the element :")
                lst.append(ele)
            insert_sort(lst,size)
        elif opt==4:
            print "Exiting....................."

main()
```

Output

```
Main Menu: Sorting
------------------
1 - Bubble Sorting
2 - Selection Sorting
3 - Insertion Sorting
4 - Quit
------------------
Enter you choice: 1
Enter the size of list-1:5
Enter the element :3
Enter the element :4
Enter the element :5
Enter the element :2
Enter the element :1

Iteration: 1
3 4 2 1 5
Iteration: 2
3 2 1 4 5
Iteration: 3
2 1 3 4 5
Iteration: 4
1 2 3 4 5
Main Menu: Sorting
------------------
```

Program No: 13

A program for searching process in a list.

Program Code

```
def linear_search(lst,size,item):
    flag=0
    for i in range(size):
        if lst[i]==item:
            flag=1
            print item," present at position ",i+1
    if flag==0:
        print item," is not present in the list"

def binary_search(lst,size,item):
    flag=0
    beg=0
    end=size-1
    while beg<end:
        mid=int((beg+end)/2)
        if item==lst[mid]:
            flag=1
            print item," present at position ",mid+1
            return
        elif lst[mid]<item:
            beg=mid+1
        else:
            end=mid-1
    if flag==0:
        print item," is not present in the list"
```

```
def main():
    while True:

        print

        print("Main Menu: Searching")
        print("------------------")
        print("1 - Linear Searching")
        print("2 - Binary Searching")
        print("3 - Quit")
        print("------------------")
        opt=input("Enter you choice: ")
        if opt==1:
            lst=[]
            size=input("enter the size of list-1:")
            for i in range(size):
                ele=input("Enter the element :")
                lst.append(ele)
            item=input("Enter the element to search:")
            linear_search(lst,size,item)
        elif opt==2:
            lst=[]
            size=input("enter the size of list-1:")
            for i in range(size):
                ele=input("Enter the element in ascending order :")
                lst.append(ele)
            item=input("Enter the element to search:")
            binary_search(lst,size,item)
        elif opt==4:
            print "Exiting....................."
            exit()
main()
```

Output

```
Main Menu: Searching
------------------
1 - Linear Searching
2 - Binary Searching
3 - Quit
------------------
Enter you choice: 1
enter the size of list-1:5
Enter the element :2
Enter the element :6
Enter the element :90
Enter the element :1
Enter the element :4
Enter the element to search:6
6  present at position  2

Main Menu: Searching
```

Program No: 14

A program to delete elements from a list (Stack Implementation)

Program Code

```
theList = list() # Creating a default list/array
# Adding element into array
def createList(theList):
    ctr = 1
    Ch = 'Y'
    while Ch == 'Y' or Ch == 'y'or Ch == 'Yes':
        print ("Enter number %d " % (ctr))
        val = float(input())
        theList.append(val) # adding number into list
        print ("Do you want to add more...<y/n>: ")
        Ch = raw_input()
        ctr += 1
        if Ch == 'N' or Ch == 'n' or Ch == 'No' or Ch == 'NO':
            break
# Delete first element
def deleteAtBeginning(theList):
    ctr = len(theList)
    if ctr > 0:
         delVal = theList.pop(0) # delete first (0th) element
         print ("Deleted value is:", delVal)

    else:
        print ("List is empty")
```

```
# Delete by value

def DeleteByValue(theList, delVal):
    Flag = 0
    for i in range(len(theList)):
        if theList[i] == delVal:
            del (theList[i])
            Flag = 1
            break
        if Flag == 0:
            print("Element does not found in list")
        else:
            print("Successfully deleted:", delVal)

# Delete at end

def deleteAtEnd(theList):

    ctr = len(theList)

    if ctr > 0:

        delVal = theList.pop() # delete last element
        print ("Deleted value is:", delVal)

    else:
        print ("List is empty")


# Show array elements
def show_element(theList):
    ctr = len(theList)
```

```
    if ctr > 0:
        print("The array is:...")
        for i in range(0, len(theList)):

            print(theList[i],)

    else:
        print ("List is empty")

Opt = 0

while (True):
    print()
    print (' M A I N M E N U')
    print ('- - - - - - - - - -')
    print ('1. Create a list')
    print ('2. Delete at beginning')

    print ('3. Delete by number/value')

    print ('4. Delete at end')

    print ('5. Show elements')

    print ('6. Quit')

    print ('- - - - - - - - - -')

    Opt = int(input( "Enter your option: "))
    print()
```

```
if (Opt == 1) :
    createList(theList)
elif (Opt == 2) :
    deleteAtBeginning(theList)
elif (Opt == 3) :
    delVal = input("Enter the alue which you want to delete: ")
    DeleteByValue(theList, delVal)
elif (Opt == 4) :
    deleteAtEnd(theList)
elif (Opt == 5) :
    show_element(theList)
elif (Opt == 6):
    break
```

Output

```
M A I N M E N U
- - - - - - - - - -
1. Create a list
2. Delete at beginning
3. Delete by number/value
4. Delete at end
5. Show elements
6. Quit
- - - - - - - - - -
Enter your option: 1
()
Enter number 1
1
Do you want to add more...<y/n>:
y
Enter number 2
4
Do you want to add more...<y/n>:
y
Enter number 3
5
Do you want to add more...<y/n>:
```

Program No: 15

A program for stack operations.

Program Code

```
Student = list() # A default stack using list() function.
top = -1 # To know the current index position in Stack.
#Adding element into a stack.
def PUSH(Student, top):
  Ch = 'Y'
  while Ch == 'Y' or Ch == 'y'or Ch == 'Yes':
    Rollno = int(input("Enter roll number: "))
    Name = raw_input("Enter name: ")
    std = (Rollno, Name) # Creating a tuple
    Student.append(std) # A tuple added into a list
    top += 1 # It check the total number of addition
    print ("Do you want to add more...<y/n>: ")
    Ch = raw_input()
    if Ch == 'N' or Ch == 'n' or Ch == 'No' or Ch == 'NO':
      break
  return top
#Removing stack elements
def POP(Student, top):
  slen = len(Student)   # Finds total elements in the stack.
  if slen <= 0:   # Checks if stack is either empty or not.
    print ("Stack is empty")
  else:
    Rollno, Name = Student.pop() # Removing top elements from Student
stack.
    top = top – 1
 print("Value deleted from stack is", Rollno, Name)
  return top
```

```
# Showing stack elements
def SHOW(Student, top):
  slen = len(Student) # Finds total elements in the Student stack.
  if slen <= 0: # Checks if stack is either empty or not.
    print ("Stack is empty")
  else:
    print("The stack elements are...")
    i = top
    while (i >= 0): # Student stack processed in reverse order.
      Rollno, Name = Student[i]
      print(Rollno, Name)
      i -= 1
while (True):
  print()
  print('S T A C K              O P E R A T I O N')
  print('- - - - - - - - - - -        - - - - -')
  print('1. Add Student Data')
  print('2. Remove Student Data')
  print('3. Display Student Data')
  print('4. Stop operation')
  Opt = input( "Enter your option: ")
  print()
  if (Opt == 1) :
    # Push operation of stack
    top = PUSH(Student, top)
  elif (Opt == 2) :
    # Pop operation of stack
    top = POP(Student, top)
  elif (Opt == 3) :
    SHOW(Student, top)
  elif (Opt == 4) :
    break
```

Output

```
S T A C K          O P E R A T I O N
- - - - - - - - - - -    - - - - -
1. Add Student Data
2. Remove Student Data
3. Display Student Data
4. Stop operation
Enter your option: 1
()
Enter roll number: 5
Enter name: Kevin
Do you want to add more...<y/n>:
y
Enter roll number: 10
Enter name: Devika
Do you want to add more...<y/n>:
n
```

Program No: 16

A program for performing Queue operations.

Program Code

```
BusQueue = list() # A default queue using list() functon.
rear = front = -1 # Initializing the queue position
#                 Inserting element into a queue.
def Insert_Q(BusQueue, rear):
    Ch = 'Y'
    while True:
        Ticketno = int(input("Enter ticket no.: "))
        Pname = raw_input("Enter passenger name: ")
        rear += 1
        Bus = (Ticketno, Pname) # Creating a tuple
        BusQueue.append(Bus) # A tuple added into a list
        print  "Do you want to add more...<y/n>: ",
        Ch = raw_input()
        if Ch == 'N' or Ch == 'n' or Ch == 'No' or Ch == 'NO' or Ch == 'no':
            break
    return rear
    #               Removing queue elements
def Delete_Q(BusQueue, rear):
    Plen = len(BusQueue)   #Finds total passengers          in bus
    if Plen <= 0:                    # Checks        if the bus is empty or
                    not.
        print ("Bus is empty")
    else:
        rear -= 1
        Ticketno, Pname = BusQueue.pop(0)
        # Removing top elements from queue.
        print("Ticket no. %d : Passenger name: %s deleted" % (Ticketno,
Pname))
    return rear
```

```
# Showing queue elements
def Show_Q(BusQueue, rear):
    front = 0
    Plen = len(BusQueue) # Finds total passengers in bus
    if Plen <= 0:
        # Checks if the bus is empty or not.
        print ("Bus is empty")
    else:
        print ("Passenger information")
        print ("="*46)
        print ("{0:^15} {1:<20}".format("Ticket No.", "Passenger Name"))
        print ("-"*46)
        while front <= rear: # Queue elements processed.
            Ticketno,Pname = BusQueue[front]
            print ("{0:^15} {1:<20}".format(Ticketno, Pname))
            front += 1
while (True):
    front = -1
    print
    print ('P A S S E N G E R O P E R A T I O N')
    print ('- - - - - - - - - - - - - - - - - - - -')
    print ('1. Insert passenger into bus')
    print ('2. Delete passenger from bus')
    print ('3. Show passenger list')
    print ('4. Exit from operation')
    Opt = int(input( "Enter your option: "))
    print
    if (Opt == 1) :
        # Insert operation of Queue - Adding element at rear of the queue
        rear = Insert_Q(BusQueue, rear)
    elif (Opt == 2) :
        # Delete operation of queue - Deleting element at front of the queue
        rear = Delete_Q(BusQueue, rear)
    elif (Opt == 3) :
```

```
elif (Opt == 3) :
        # Traversing/Showing queue element
        Show_Q(BusQueue, rear)
    elif (Opt == 4) :
        break
    raw_input()
```

Output

```
P A S S E N G E R              O P E R A T I O N
- - - - - - - - - - - - - - - - - - - -
1. Insert passenger into bus
2. Delete passenger from bus
3. Show passenger list
4. Exit from operation
Enter your option: 1

Enter ticket no.: 25
Enter passenger name: thomas
Do you want to add more...<y/n>:  y
Enter ticket no.: 30
Enter passenger name: mathew
Do you want to add more...<y/n>:  n


P A S S E N G E R              O P E R A T I O N
- - - - - - - - - - - - - - - - - - - -
1. Insert passenger into bus
2. Delete passenger from bus
3. Show passenger list
4. Exit from operation
Enter your option: 3

Passenger information
==============================================
 Ticket No.   Passenger Name
----------------------------------------------
    25       thomas
    30       mathew
```

Program No: 17

A program to count the number of words

Program Code

```
print ("This program calculates the number of words in a sentence")
print
p=raw_input("Enter a sentence: ")
words = str.split(p)

count = len(words)
print count
print (("The total word count is:"),count)
```

Output

This program calculates the number of words in a sentence

Enter a sentence: this is one example

4

('The total word count is:', 4)

Program No: 18

A program to reads a characters from the file one by one. All lower case characters get stored inside the file LOWER, all uppercase characters get stored inside the file UPPER and all other characters get stored inside file OTHERS.

Program Code

```
read_file=open("data.txt","w")
read_file.write("Hello World in the new File 2016 \n")
()
read_file=open("data.txt","r")
string=''
string=read_file.read()
read_file.close()
upper=''
lower=''
other=''
print string
for letter  in string:
    if letter.isalpha():
        if letter.isupper():
            upper+=letter
        elif letter.islower():
            lower+=letter
    else:
        other+=letter
print 'Upper=',upper
print 'Lower=',lower
print 'Other=',other
upper_file=open("upper.txt",'wb')
lower_file=open("lower.txt",'wb')
other_file=openread_file.close ('other.txt','wb')
upper_file.write(upper)
lower_file.write(lower)
other_file.write(other)
upper_file.close()
lower_file.close()
other_file.close()
```

Output

Hello World in the new File 2016

Upper= HWF

Lower= elloorldinthenewile

Other= 2016

Program No: 19

Write a file handling program using binary file create an instance of Class Student onto a file namely student.log. Data members are Roll No, Name and Marks

Student Data Entry

1. Add student record
2. Display the student record based on given Roll No

Program Code

```
import pickle
class Student():
  def __init__(self,roll=0,name='',marks=0):
    self.rollno=roll
    self.name=name
    self.marks=marks
  def add(self):
    self.rollno=int(input("Enter Roll No:"))
    self.name=(raw_input("Enter Name:")).upper()
    self.marks=int(raw_input("Enter Marks:"))
  def display(self):
    print 'Roll No:\t',self.rollno
    print 'Name:\t',self.name
    print 'Marks:\t',self.marks
def save(obj):
  f=open('student_data.log','ab+')
  pickle.dump(obj,f)
  f.close()
def open_data():
  f=open('student_data.log','rb+')
  data_list=[]
  try:
    while True:
      obj=pickle.load(f)
      data_list.append(obj)
  except EOFError:
```

```
 f.close()
   print '\t\t\t\t\t<DATA READ>'
  return data_list
def main():
  print '_'*80
  print "\t\tSTUDENT MARK DATABASE"
  print '_'*80
  print '\t1.Add New Record'
  print '\t2.Search by Roll Number'
  print '\t3.Search by Name'
  print '\t4.Display All Records'
  print '\t5.Exit'
  print '-'*50
  print

  option=int(raw_input("Enter Choice: "))
  print '_'*75
  if int(option)==1:
    obj=Student()
    obj.add()
    save(obj)

 elif int(option)==2:
    no=int(raw_input("Enter Student Roll No to Search: "))
    s=0
    data_list=open_data()
    for x in data_list:
      if x.rollno==no:
        x.display()
        s=1
    if s!=1:
      print "\t<RECORD NOT FOUND>"
  elif int(option)==3:
    name=raw_input("Enter Student Name to Search: ")
    s=0
    data_list=open_data()
```

```
for x in data_list:
      if x.name==name.upper():
        x.display()
        s=1
    if s!=1:
      print "\t<RECORD NOT FOUND>"
  elif int(option)==4:
    print 'DISPLAYING ALL RECORDS'
    print '.........................................'
    data_list=open_data()
    print
    for x in data_list:
      x.display()
      print '-'*50
      print
      s=1
    if s!=1:
      print "\t<NO RECORD FOUND>"
  elif int(option)==5:
    exit()
  else:
    print "\t<WRONG CHOICE>"
  raw_input()

while True:
  main()
```

Output

```
_____________________________________________________
_____________________________________________
                    STUDENT MARK DATABASE
_____________________________________________________
_____________________________________________
        1.Add New Record
        2.Search by Roll Number
        3.Search by Name
        4.Display All Records
        5.Exit
-----------------------------------------------

Enter Choice: 1
_____________________________________________________
_____________________________________
Enter Roll No: 1
Enter Name: Alex
Enter Marks: 25
```

Enter Choice: 4

DISPLAYING ALL RECORDS

.....

<DATA READ>

Roll No: 23

Name: RAHUL

Marks: 23

Roll No: 25

Name: YUVRAJ

Marks: 100

Roll No: 1

Name: ALEX

Marks: 25

Program No: 20

A program to multiply two given matrix.

Program Code

```
import os
# Program to multiply two matrices
# using nested loops
# 3x3 matrix
X = [  [5,4,6],     [2 ,5,8],      [3 ,1,4]]
# 3x4 matrix
Y = [[8,7,3,4],    [3,5,3,4],    [5,6,3,4]]
# result is 3x4
print len(Y)
print len(Y[0])
result = [[0,0,0,0],         [0,0,0,0],          [0,0,0,0]  ]
# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
      # iterate through rows of Y
      for k in range(len(Y)):
         result[i][j] += X[i][k] * Y[k][j]
for r in result:
   print(r)
raw_input()
```

Output

3

4

[82, 91, 45, 60]

[71, 87, 45, 60]

[47, 50, 24, 32]

PART II

SQL

Structured Query Language

Question No: 1

Write SQL command for (a) to (f) and write output of SQL command given in (g) with the help of table shown below :

Relation: Library

S.no.	Title	Author	Type	Pub	Qty	Price
1	Data Structure	Lipschutz	DS	Mcgraw	4	217
2	Computer Studies	French	FND	Galgotia	2	75
3	Advanced Pascal	Schildt	PROG	Mcgraw	4	350
4	Dbase Dummies	Palmer	DBMS	PustakM	5	130
5	Mastering C++	Guerewish	PROG	BPB	3	295
6	Guide Network	Freed	NET	ZPress	3	200
7	Mastering Foxpro	Seigal	DBMS	BPB	2	135
8	DOS Guiede	Norton	OS	PHI	3	175
9	Basic of Beginner	Morton	PROG	BPB	3	40
10	Mastering Window	Cowart	OS	BPB	1	225

a) Select all the PROG type published by BPB from Library.

b) Display a list of all books with Price more than 130 and sorted by Qty.

c) Display all books sorted by Price in ascending order.

d) Display all report, listing books title, current value and misplacement charges for each book in above table. Calculate the misplacement charges for all books price*1.25.

e) Count the number of books in above table. Fill all the column with values.

f) Insert a new book in Library table. Fill all column with valus.

g) Give the output of following SQL command on the basis of table Library

1. MIN(Price) from Library where Price<150;
2. Select AVG(Price) from Library where Qty<3;
3. Select COUNT(DISTINCT Pub) from Library;

Queries:

a) SELECT Type FROM Library WHERE(Type="PROG"AND Pub="BPB");

b) SELECT Title „BOOKS", FROM Library WHERE Price>130 ORDER BY Qty;

c) SELECT Title „BOOKS", FROM Library ORDER BY Price ASC;

d) SELECT Title „BOOKS", Price "Current Value", Price*1.25"Misplacement Charges" FROM Library;

e) SELECT Count(Title) FROM Library WHERE Pub="PHI";

f) INSERT INTO Library VALUES("Exploring C " , " Yashwant " , " PROG" , " BPB " , 3 , 230) ;

g)

1. 1.40
2. 2.145
3. 3.6

Question No: 2

Write SQL command for (a) to (j) with the help of table shown below :

Relation : Employee

EMPCODE	ENAME	DEPT	DESIG	BPAY
101	Vishal	EDP	System Analyst	20000
102	Joe	Marketing	Manager	15000
103	Resma	Accounts	Manager	15000
104	Joe	EDP	Programmer	12000
105	Vivek	Marketing	Executive	6000
106	Ram	Accounts	Clerk	3000
107	Rajeev	EDP	Programmer	12000
108	Peter	Accounts	Accountant	15000
109	Rahim	Marketing	Executive	6000
110	Neethu	EDP	Manager	25000

a) Display the details of all managers.

b) List the name and designation of employees in EDP department.

c) List the details of employees whose salaries fall in the range of 10000 and 20000.

d) Display the departments in the company.

e) Display the department and total salary spent in each department.

f) Display the names of employees starting with the letter 'R'.

g) List down the details of managers who are getting salary of 20000 or more.

h) Display lowest salary in the EDP department.

i) Add new column in the employee table to store the DA amount.

j) Update the new column with 36% of the basic pay.

Queries:

a) SELECT * FROM Employee WHERE DESIG = “Manager‟;

b) SELECT ENAME , DESIG FROM Employee WHERE DEPT = “EDP‟;

c) SELECT * FROM EMPLOYEE WHERE BPAY BETWEEN 10000 AND 20000;

d) SELECT DISTINCT DEPT FROM EMPLOYEE;

e) SELECT DEPT, SUM(BPAY) FROM EMPLOYEE GROUP BY(DEPT);

f) SELECT ENAME FROM EMPLOYEE WHERE ENAME LIKE “R%‟;

g) SELECT * FROM EMPLOYEE WHERE DESIG=‟‟MANAGER” AND BPAY>=20000;

h) SELECT MIN(BPAY) FROM EMPLOYEE WHERE DEPT =‟‟EDP‟;

i) ALTER TABLE EMPLOYEE ADD(DA INTEGER);

j) UPDATE EMPLOYEE SET DA = BPAY * 0.36;