Binary File Program

Q.1. Write a function addRec() to create a binary file student.dat and a single record for the structure (roll, name, mark) into the binary file

Ans:
```
import pickle
def addRec():
    fobj = open('student.dat', 'wb')
    roll = int(input("Enter roll number:"))
    name= input("Enter name: ")
    mark= int(input("Enter mark: "))
    rec=[roll,name,mark]
    pickle.dump(rec, fobj)
    fobj.close()

addRec():
```

Q.2 Write a function readRec() to read record of structure (roll, name, mark) and display the record from the file,

Ans:
```
import pickle
def readRec():
    fobj = open('student.dat', 'rb')
    rec= pickle.load(fobj)
    print (rec[0], rec[1], rec[2])
    print (rec)
    fobj. close()
readRec()
```

Q, 3. Write a program to create and write a list structure in binary file data.dat. by writing a function addList(),

Ans:
```
import pickle
def addList ():
    fobj = ('data.dat', 'wb')
    list1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
    pickle.dump (list1, fobj)
    print (" Binary file is created and
                a list of numbers is added to
                binary file")

    fobj. close()

addList ()
```

Q, 4, Write a function readData () to read structure-list from binary file data-dat and display it,

Ans:
```
import pickle
def readData ():
    fobj = open ('data.dat', 'rb')
    l = pickle.load (fobj)
    print ( l[0], l[1], l[2], l[3], l[4], l[5],
            l[6], l[7], l[8], l[9])
    print (l)
    fobj. close ()
readData ():
```

Q.5, write a function writeRec() to create a binary file student2.dat and write multiple records for structure (roll, name, mark) in the binary file,

Ans:
```
import pickle
def writeRec( ):
    f = open('student2.dat', 'wb')
    while True:
        roll = int (input (" Enter roll : "))
        name = input (" Enter name : ")
        mark = int (input (" Enter mark: "))
        rec = [roll, name, mark]
        pickle.dump (rec, f)
        ch = input (" want to give another
                    record (Y/N): ")
        if ch.upper () == 'N' :
            break            ↘ [ if ch in 'Nn' :
                                    break ]
    print (" record added")
    f.close ()
writeRecord ()
```

Q.6, A binary file 'student2.dat' has structure (roll, name, mark) write a function displayRec() that would read content/record of the file 'student2.dat' and display details of students,

Ans:
```
import pickle
def displayRec ():
    fobj = open ('student 2.dat', 'rb')
    while True:
        try:
            rec = pickle.load (fobj)
            print ( rec[0], rec[1] , rec[2])
            # print (rec) - for list view
        except EOFError:
    f.obj.close()  break
displayRec ()
```

Q.7, A binary file student2.dat has structure (roll, name, mark), write function searchRec() in python that would read contents of the file student2.dat and display the details of those students who has got above 75 mark, Also display number of students scoring mark above 75

Ans:
```
import pickle
def searchRec():
        fobj = open('student2.dat', 'rb')
         count = 0
         While True:
             try:
                rec = pickle.load(fobj)
                if rec[2] > 75:
                        count += 1
                        print(rec[0], rec[1], rec[2])

             except EOFError:
                   break
          fobj.close()
          print("Number of students got over 75:", count)
searchRec()
```

Q.8. Anita Bose is a programmer, who has recently given a task to write a python code to perform the following binary file operations with the help of two following functions/modules:

(a) AddBook () — to create a binary file called BOOK.DAT — containing book — information - Book-No, Book_Name, Author and Price,

(b) GetBook (Author) - that accepts the Author name as parameter and counts and returns the number of books by the given Author and total amount of price of the given Author books — are stored in binary file BOOK-DAT

Ans:
```
import pickle
def AddBook():
    fobj = open("BOOK.DAT", "ab")
    while True:
        Book_No = int(input("Enter Book Number:"))
        Book_Name = input("Enter Book Name:")
        Author = input("Enter Author Name:")
        Price = int(input("Enter price of book:"))
        rec = [Book_No, Book_Name, Author, Price]
        pickle.dump(rec, fobj)
        ch = input("enter more record(Y/N):")
        if ch in 'Nn':
            break
    print("Record Added")
    fobj.close()
```

```
def GetBook(Author):
    fobj = open('BOOK.DAT', 'rb')
    count=0
    total_price= 0
    while True:
        try:
            rec = pickle.load(fobj)
            if Author == rec[2]:
                count = count+1
                total_price = total_price +rec[3]
        except EOFError:
            break
    fobj.close()
    print("The number of Author:", Author, "is", count)
    print(" The total price of the Author's book:",
                                            total_price)
AddBook()
GetBook("Khushbant Sing")
```

Q.9. Anirban Sharma is a programmer, who has recently been given a task to write a python code to perform the following binary file operations with the help of two user defined functions/modules:

(a) AddStudents () to create a binary file called STUDENT.DAT containing student information - roll number, name and marks (out of 100) of each student,

(b) GetStudents () to display the name and percentage of those students who have a percentage greater than 75; In case there is no student having percentage >75, the function displays an appropriate message. The function should also display the average percentage,

Ans:
```
import pickle
def AddStudents ():
    F= open ("STUDENT.DAT", "wb")
    while True:
        Rno = int(input ("Rno: "))
        Name= input ("Name: ")
        Percent= float (input ("Percent: "))
        L = [Rno, Name, Percent]
        pickle. dump (L,F)
        choice = input ("enter more (Y/N): ")
        if choice in'nN':
            break
    F.close()
```

```
def GetStudents ():
    Total =0
    countrec=0
    countabove75=0
    with open("STUDENT.DAT", "rb") as F:
        while True:
            try:
                R= pickle.load(F)
                countrec +=1
                Total += R[2]
                if R[2] > 75:
                    print (R[1], "has percentage:",
                                R[2])
                    countabove75 +=1
            except EOFError:
                break
        if countabove75= =0:
            print ("There is no student got above
                     75 percentage ")
    average= Total/countrec
    print(" average percentage of class :", average)

AddStudents ()
GetStudents ()
```

Q.10. Rakesh Mishra is an employee, your teacher has given you a task to write a python code to perform the following binary file operations for the employee record with the help of two user defined functions/modules:

(a) AddEmployees () to create a binary file called EMPLOYEE.DAT containing employee information – employee id, name, salary, department of each employee

(b) GetEmployees () to count display name, id, salary, department of those employees who have salary more than 15000, In case there is no employee earning more than 15000, the function displays an appropriate message, the function should also display the average salary of employees

Ans:
```
import piekle
def AddEmployees ():
    fobj = open(" EMPLOYEE.DAT", "ab")
    while True:
        emp-id = int (input ("Enter employee id : "))
        name = input (" Enter employee name : ")
        salary = eval (input (" Enter salary : "))
        dept = input ("Enter salary : ")
        rec = [emp-id, name, salary, dept]
        piekle.dump (rec, fobj)
        ch = input ("Enter more record (Y/N)? : ")
        if ch in 'Nn' :
            break
    print(" Record Added")
    fobj. close ()
```

```
def GetEmployees():
    fobj = open("EMPLOYEE.DAT", "rb")
    count = count_sal = 0
    total_sal = 0
    while True:
        try:
            rec = pickle.load(fobj)
            count += 1
            total_sal += rec[2]
            if ree[2] > 15000:
                print(rec[0], rec[1], rec[2], rec[3])
                count_sal += 1
        except EOFError:
                break
    if count_sal == 0:
        print("There is no employee earning more
                than 15000")
    avg = total_sal/count
    print("The average salary of employees :", avg)
    print(" Total number of employee earning more
            than 15000: ", count_sal)

Add Employees ()
Get Employees ()
```