SAHODAYA PRE BOARD EXAMINATION (2024-25)

CLASS – XII

Sub.: COMPUTER SCIENCE (083)

MARKING SCHEME (SET – 1)

	SECTION – A	
1.	False *(1 mark for correct answer)*	**[1]**
2.	(A) blue#green#red *(1 mark for correct answer)*	**[1]**
3.	(A) True *(1 mark for correct answer)*	**[1]**
4.	(A) (“CBS”, “E”, “ Examination 2025”) *(1 mark for correct answer)*	**[1]**
5.	(C) 'u@opt' (1 mark for correct answer)	**[1]**
6.	(B) [12, 13, 3, 4] *(1 mark for correct answer)*	**[1]**
7.	(C) {28:'Feb',30:'Apr'}+{31:'Jan'} *(1 mark for correct answer)*	**[1]**
8.	(C) It will give “ValueError”. *(1 mark for correct answer)*	**[1]**
9.	(A) 4 *(1 mark for correct answer)*	**[1]**
10.	(D) All of the above. *(1 mark for correct answer)*	**[1]**
11.	False *(1 mark for correct answer)*	**[1]**
12.	(B) –2 *(1 mark for correct answer)*	**[1]**
13.	ALTER *(1 mark for correct answer)*	**[1]**
14.	(D) Details of all employees whose names contains ‘O’ in the second place and is of 5 characters. *(1 mark for correct answer)*	**[1]**
15.	(B) VARCHAR *(1 mark for correct answer)*	**[1]**
16.	(D) NULL values *(1 mark for correct answer)*	**[1]**
17.	(C) IMAP *(1 mark for correct answer)*	**[1]**
18.	(C) Repeater *(1 mark for correct answer)*	**[1]**
19.	Circuit switching *(1 mark for correct answer)*	**[1]**
20.	(B) Both A and R are true and R is not the correct explanation of A. *(1 mark for correct answer)*	**[1]**
21.	(A) Both A and R are true and R is the correct explanation for A. *(1 mark for correct answer)*	**[1]**

	SECTION – B	
22.	**formal parameter** — the identifier used in a method to stand for the value that is passed into the method by a caller. Also known as **Parameters.** For example, amount is a formal parameter of processDeposit **actual parameter** — the actual value that is passed into the method by a caller. For example, the 200 used when processDeposit is called is an actual parameter. actual parameters are often called **arguments.**	**[2]**
23.	I) + is Arithmetic, >= is Relational, AND is Logical II) +, >=, AND (Arithmatic, relational, logical) Python will always evaluate the arithmetic operators first (** is highest, then multiplication/division, then addition/subtraction). Next comes the relational operators. Finally, the logical operators are done last. *(1 mark for identifying any two types of operators)* *(1 mark for writing the correct hierarchy of operators)*	**[2]**
24.	I. A). L1.append(45) OR B) L2 . insert (5 , 15) *(1 mark for correct answer)* II. A) L2.pop() OR B) max(L1) *(1 mark for correct answer)*	**[2]**
25.	(B) *30#40#50#* *(½ mark for all correct numbers and ½ mark for #)* Maximum value assigned to FROM and TO variables is 3 and 4 respectively. *(½* x *2 = 1 Mark for ecah correct value of maximum)*	**[2]**

26.	n = **int**(input("Enter N: ")) #int() missing
	sum = **0** #sum = 0
	if n < 0:
	    for i in range(2 * n, **n+1**): #n+1 as n is included
	        sum += i
	else:
	    for i in range(n, 2 * n+1):
	        sum += **i** #i instead of I
	print("Sum =", sum)
	
	(1/2 for each correction done)
		[2]

27.	(I) A) CHECK OR B) UNIQUE or PRIMARY KEY (II) A) ALTER TABLE Persons ADD PRIMARY KEY(P_Id); OR B) The given statement upon execution drops the PRIMARY KEY constraint in table Persons.	**[2]**

28. **[2]**

	Circuit Switching	**Packet Switching**
Connection Establishment	A dedicated path is established for the entire duration of the call.	Data is sent in packets that are routed independently, with no dedicated path.
Resource Utilization	Resources are reserved for the entire connection, even during idle periods.	Resources are used only when packets are transmitted, allowing for efficient sharing.
Data Transmission	Data is transmitted in a continuous stream.	Data is broken into packets that can arrive out of order.
Scalability	Less scalable; establishing multiple circuits can overwhelm the network.	Highly scalable; can easily accommodate many users sharing the same network resources.

(2 marks for correct differences)

OR

ISP stands for **Internet Service Provider**
Airtel, Reliance Jio Fiber, BSNL, etc.
(1 mark for full form)
(1 mark for any two ISPs)

SECTION – C

29. **[3]**

```
def RevText():
      f=open("Story.txt","r")
      y=f.read()
      Words=y.split()
      for word in Words:
            if word[0]=='I' or word[0]=='i':
                print(word[::-1],end=" ")
            else:
                 print(word,end=" ")
       f.close()
```

(½ mark for correct function header)
(½ mark for correctly opening the file)
(½ mark for correctly reading from the file)
(½ mark for splitting the text into words)
(1 mark for correctly displaying the desired words)

OR

```
def countmy():
      f=open("Data.txt","r")
      count=0
      y=f.read()
      Words=y.split()
      for word in Words:
            if word=='my' or word=='My':
                count=count+1
       print("my occurs ",count," times")
       f.close()
```

(½ mark for correct function header)
(½ mark for correctly opening the file)
(½ mark for correctly reading from the file)
(½ mark for splitting the text into words)
(1 mark for correctly displaying the desired words)

30. (A) **[3]**

(I)

```
def push_item(ItemStack, new_item):
        ItemStack.append(new_item)
```

(II)

```
def pop_item(ItemStack):
        if not ItemStack:
                print("Underflow")
        else:
                return(ItemStack.pop())
```

(III)

```
def peep(ItemStack):
        if not ItemStack:
                print("None")
        else:
                print(ItemStack[-1])
```

(3x1 mark for correct function body; No marks for any function header as it was a part of the question)

OR

(B)

```
def push_nums(N):
        Numbers = []
        for num in N:
                if num>0 and num % 2 == 0:
                        Numbers.append(num)
        return Numbers
VALUES = [ ]
ans= ‘y’
while ans== ‘y’:
        VALUES.append(int(input("Enter an integer: ")))
        ans=input(“any more y/n “)
Numbers = push_nums(VALUES)

def pop_num():
        if not Numbers:
                print("Empty")
        else:
                print(Numbers.pop())
pop_num()

def disp_num():
        if not Numbers:
                print("None")
        else:
                print(Numbers)
disp_num()
```

(1/2 mark for identifying numbers)
(1/2 mark for correctly adding data to stack)
(1/2 mark for correctly poping data on the stack and 1/2 mark for checking condition)

(1/2 mark for correctly displaying the data with none)
(1/2 mark for function call statements)

31. #BSE *3135 **[3]**

(1.5 mark for #BSE * *and 1.5 mark for* 3135*)*

OR

New Delhi Beijing Washington DCOk LondonOk

(1 mark for New Delhi Beijing
1 mark for Washington DCOk
1 mark for LondonOk)

SECTION – D

32. (A) **[4]**

(I) SELECT ITEMNAME,PRICE,SECTION FROM DRESS ORDER BY PRICE DESC, SECTION;
(II) SELECT ITEMNAME, PRICE-(PRICE*0.1) AS ‘netprice’ FROM DRESS;
(III) SELECT * FROM DRESS WHERE ITEMNAME LIKE ‘%e’ ;
(IV) SELECT SECTION, MAX(PRICE),MIN(PRICE) FROM DRESS GROUP BY SECTION;

OR

(B)

(I)
```
ITEMNAME      SIZE
Pant          36
Shirt         42
Jeans         44
```
(II) **COUNT(DISTINCT SECTION)** 2

(III)
```
SIZE          PRICE
40            1200
36            6000
36            2900
34            3400
32            2600
```
(IV)
```
DCODE         PRICE
S002          3000
S005          2500
S007          2300
```

33. (I) **[4]**
```
def ADD( ):
        import csv
        field = ["RollNo", "Name" , "Percentage"]
        f = open("student.csv" , 'w')
        d=csv.writer(f)
        d.writerow(field)
        ch='y'
        while ch=='y' or ch=='Y':
                roll=int(input("Enter the roll number: "))
                name= input("Enter the Name: ")
                percent=float(input("Enter the percentage of marks: "))
                rec=[roll, name, percent]
                d.writerow(rec)
                ch=input("Enter more record??(Y/N)")
            f.close()
ADD()
(II) def Display( ):
        import csv
        f = open("student.csv " , "r")
        d = csv.reader(f)
        next(f)         #to skip header row
        for row in d:
            if int(row[2])>90:
              print(row)
        f.close()
   Display()
```

(1$\frac{1}{2}$ for each correct definition of function
½ for importing csv module
½ for calling the functions)

34. **[4]**

```
I)    SELECT DName, Department, Charges
      FROM DOCTOR DO, DEPT DE
      WHERE DO.DId=DE.DId;
II)   SELECT SUM(Charges) FROM DEPT
      GROUP BY Department;
III)  UPDATE DEPT
      SET Charges=Charges*0.1
      WHERE Department='Neurology';
IV)   (A) SELECT * FROM DOCTOR
      WHERE Age BETWEEN 40 and 50 and Gender='M';
```

OR

```
      (B) SELECT Gender, Count(*) FROM DOCTOR
      GROUP BY Gender;
```

(4x1 mark for each correct query)

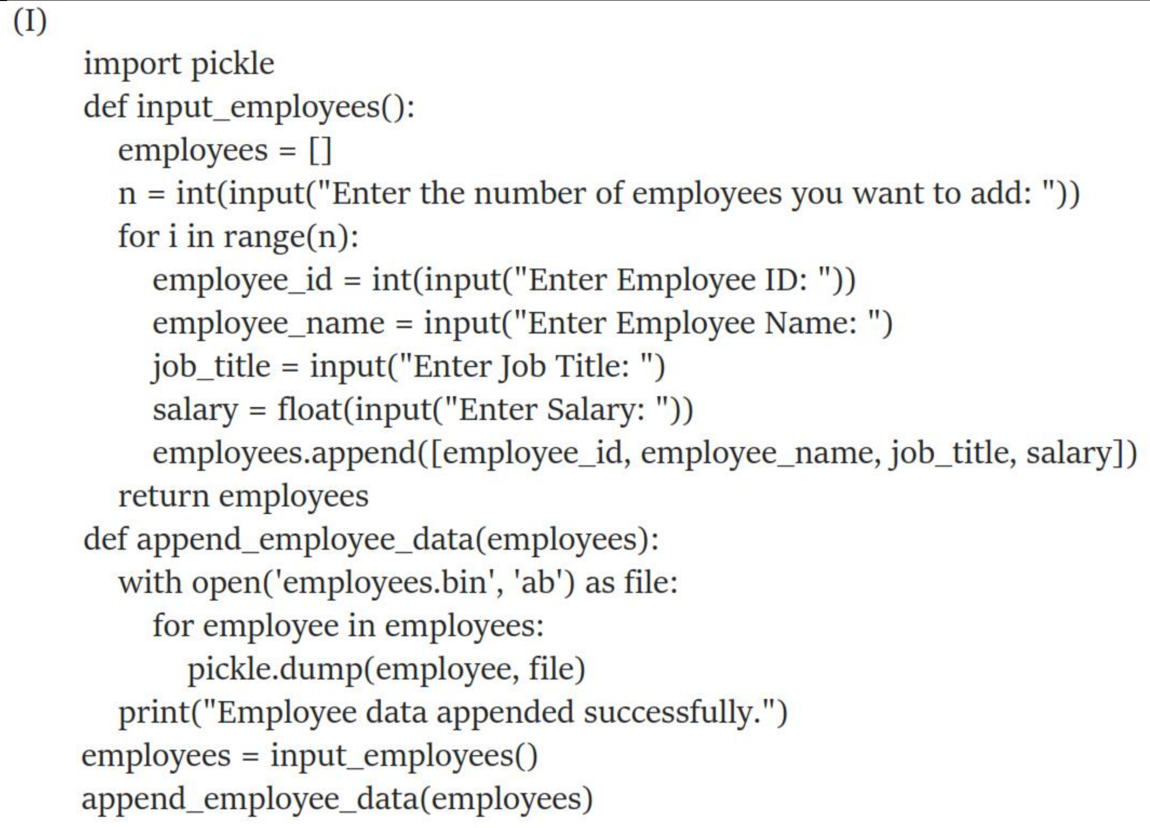
35. **[4]**

```
import mysql.connector as mycon
def AddAndShowProducts():
    mydb = mycon.connect(host="localhost", user="admin",
   passwd="Secret123", database="SHOPDB")
    mycur = mydb.cursor()
    productID = int(input("Enter Product ID: "))
    productName = input("Enter Product Name: ")
    cost = float(input("Enter Product Cost: "))
    stock = int(input("Enter Stock Quantity: "))
    query = "INSERT INTO PRODUCTS VALUES ({}, '{}', {}, {})"
    query = query.format(productID, productName, cost, stock)
    mycur.execute(query)
    mydb.commit()
    print("Product added successfully.")
    mycur.execute("SELECT * FROM PRODUCTS WHERE cost < 50")
    for rec in mycur:
        print(rec)
    mycur.close()
    mydb.close()
    print("Database connection closed.")
AddAndShowProducts()
```

(½ mark for correctly importing the connector object)
(½ mark for correctly creating the connection object)
(½ mark for correctly creating the cursor object)
(½ mark for correctly inputting the data)
(½ mark for correct creation of first query)
(½ mark for correctly executing the first query with commit)
(½ mark for correctly executing the second query)
(½ mark for correctly displaying the data)

SECTION – E

36. **[5]**


(I)

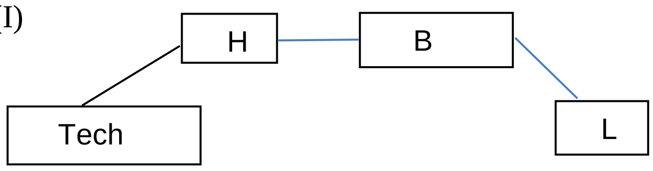
```
import pickle
def input_employees():
    employees = []
    n = int(input("Enter the number of employees you want to add: "))
    for i in range(n):
        employee_id = int(input("Enter Employee ID: "))
        employee_name = input("Enter Employee Name: ")
        job_title = input("Enter Job Title: ")
        salary = float(input("Enter Salary: "))
        employees.append([employee_id, employee_name, job_title, salary])
    return employees
def append_employee_data(employees):
    with open('employees.bin', 'ab') as file:
        for employee in employees:
            pickle.dump(employee, file)
    print("Employee data appended successfully.")
employees = input_employees()
append_employee_data(employees)
```

(II)

```
import pickle
def update_senior_developer():
    updated_employees = []
    try:
        with open('employees.bin', 'rb') as file:
            while True:
                try:
                    employee = pickle.load(file)
                    if employee[3] > 100000:
                        employee[2] = 'Senior Developer'
                    updated_employees.append(employee)
                except EOFError:
                    break
    except FileNotFoundError:
        print("No employee data found. Please add employees first.")
        return
    with open('employees.bin', 'wb') as file:
        for employee in updated_employees:
            pickle.dump(employee, file)
    print("Employees updated to Senior Developer where applicable.")
update_senior_developer()
```

(III)

```
import pickle
def display_non_senior_developers():
    try:
        with open('employees.bin', 'rb') as file:
            while True:
                try:
                    employee = pickle.load(file)
```

	if employee[2] != 'Senior Developer': print("Employee ID:", employee[0]) print("Employee Name:", employee[1]) print("Job Title:", employee[2]) print("Salary:", employee[3]) print("--------------------") except EOFError: break # End of file reached except FileNotFoundError: print("No employee data found. Please add employees first.") print("\nNon-Senior Developers:") display_non_senior_developers() (1/2 mark of import pickle) (1/2 mark for input) (1/2 mark for opening file in append mode and 1/2 mark for using dump) (1/2 mark for opening file in read mode and 1/2 mark for using load) (1 mark for checking the condition and updating the value) (1 mark for checking the condition and displaying data correctly)	
37.	(I)  *(1 mark for correct answer)* (II) Switch/ Hub *(1 mark for correct answer)* (III) HR Center as it is having maximum computer. *(1 mark for correct answer)* (IV) WAN as the distance is more. *(1 mark for correct answer)* (V) (A) Firewall **OR** (B) Repeater is not required in the network as the distance is not exceeding 80 meters. *(1 mark for correct answer)*	**[5]**

* * * * *