Class 12 – Computer Science (083)

Pre-Board 1-Examination 2023–24 – Solved Paper

SECTION A

1. a) key

2. d) Data Redundancy

3. a) -2

4. c) [55, 44, 33, 22, 11]

5. b) query

6. b) del D1["Red"]

7. b) >>>a % b % c + 1.0

8. c) >>>t + (60,)

9. b) t = (1,)

10. b) The shuffled list x with the elements mixed up

11. b) x is 50; Changed global x to 2; Value of x is 2

12. true

13. b) write into the buffer

14. b) f.readlines()

15. a) 48

16. a) executes()

17. d) A is False but R is True

18. a) Both A and R are true and R is the correct explanation

SECTION B

19. Find the error and correct the code

```
mul = 3
value = 10
for i in range(1, 6,1):
   if (value % mul == 0):
      print(value * mul)
   else:
      print(value + mul)
```

20. Advantages of DBMS

- Data Redundancy reduced
- Data Consistency
- Security & Integrity

- Backup & Recovery

OR

- **Domain:** Set of valid values for a column. Example: Age domain = 1 to 100
- **Pool:** Resource group (like connection pool)

21. Prime number code (corrected):

```
def prime():
    n = int(input("Enter number to check :: "))
    for i in range(2, n // 2):
        if n % i == 0:
            print("Number is not prime\n")
            break
    else:
        print("Number is prime\n")
```

The corrections are:

- In the line, n=int(input("Enter number to check :: ")) , the underlined bracket was missing.
- In the line, if n%i = = 0 : , the underlined equal to symbol was missing.
- In the line, break , the indentation of break was incorrect.
- In the line, print("Number is prime \n ") , the quotes inside the print( ) did not match.

22. Attribute vs Tuple

- **Attribute**: A column (e.g., Name)
- **Tuple**: A row (e.g., ('Raj', 25, 'Delhi'))

23. a) for is for known iterations, while for unknown
b) break exits loop early

24. Output

['raj', 'seema', 'john', 'smith']

OR

22 # 40 # 9 # 13 #

25. COUNT() vs COUNT(*)

- COUNT(column) ignores NULLs
- COUNT(*) counts all rows

OR

- DDL: ALTER, DROP
- DML: INSERT, UPDATE

SECTION- C

26. a)

ACode	Name	Type	City
A01	Amrita	Savings	Delhi
A01	Amrita	Savings	Nagpur
A02	Parthodas	Current	Mumbai

b)SQL Queries (TECH_COURSE assumed):

i.

DISTINCT TID
101
NULL
102
104
103

ii.

TID	COUNT(*)	MIN(FEES)
101	2	12000

iii.

CNAME
Digital marketing
Mobile Application Development

iv.

AVG(FEES)
15500.00

27. COUNTLINES()

```
def COUNTLINES():
        count = 0
        with open("TESTFILE.TXT", "r") as f:
        for line in f:
            if line[0].lower() not in "aeiou":
                count += 1
print("The number of lines not starting with any vowel –", count)
COUNTLINES()
```

OR

```
def ETCount():
    count_e = 0
    count_t = 0
    try:
        with open("TESTFILE.TXT", "r") as file:
            content = file.read()  # Read the entire file content
            for char in content:
                if char.lower() == 'e':
                    count_e += 1
                elif char.lower() == 't':
                    count_t += 1
        print(f"Count of 'E' or 'e': {count_e}")
        print(f"Count of 'T' or 't': {count_t}")
    except FileNotFoundError:
        print("Error: 'TESTFILE.TXT' not found. Please ensure the file
          exists in the same directory.")
    except Exception as e:
        print(f"An error occurred: {e}")
ETCount()
```

28. SQL (SPORTS)

i. SELECT GAME1, GAME2 FROM SPORTS WHERE NAME LIKE 'A%';

ii. ALTER TABLE SPORTS ADD MARKS INT;

iii. UPDATE SPORTS SET MARKS = 200 WHERE GRADE1 IN ('A','B') AND GRADE2 IN ('A','B');

iv. SELECT * FROM SPORTS ORDER BY NAME;

b. UPDATE SPORTS SET GRADE1 = 'A' WHERE STUDNO = 13;

29. Word search index in sentence

```
def search_word(sentence, word):
    result = {}
    index = 0
    while True:
        index = sentence.find(word, index)
        if index == -1:
            break
        result[index] = word
        index += len(word)
    print(result)
search_word("He is such a boy who is very good in studies.", "is")
```

30. Stack operations – Goa filter

```
status = []
def Push_element(customers):
    for c in customers:
```

```
        if c[2].lower() == "goa":
            status.append([c[0], c[1]])
def Pop_element():
    while len(status)!=0:
        print(status.pop())
    print("Stack Empty")
```

OR

```
list1=[12, 13, 34, 56, 21, 79, 98, 22, 35, 38]
def func_push (stack, num):
    stack.append(num)#it appends the received value to the stack
def func_pop (stack):
    if stack != []:
        return stack.pop() #return the topmost value if stack is not empty
    else:
        return None

my_Stack = []
for i in list1:#i represents each value of list1 at a time
    if i% 2==0: #check for even numbers
        func_push(my_Stack, i)#if value is even, it is pushed in stack
while True:
    if my_Stack!=[]: #check if the stack is empty or not
        print(func_pop(my_Stack), end="") #if not empty, then pop out values
    else:
        break
```

SECTION -D

31. SQL Queries

i. SELECT TNAME, CITY FROM TRAINER;

ii. SELECT TNAME, CITY, SALARY FROM TRAINER ORDER BY HIREDATE DESC;

iii. SELECT * FROM TRAINER WHERE TNAME LIKE '%A';

iv. SELECT * FROM COURSE ORDER BY STARTDATE;

v. SELECT CNAME, FEES FROM COURSE WHERE FEES BETWEEN 10000 AND 15000;

32. a) Output of code

105#6#

b) Remove odd index characters

```
def remove_odd(s):
    return ''.join(s[i] for i in range(len(s)) if i % 2 == 0)
```

OR

a) Output of transformation code:

sELCcME&Cc

b) Delete 4th word

```
def delete_fourth_word():
    with open("school.txt", "r") as f:
        lines = f.readlines()
    with open("school.txt", "w") as f:
        for line in lines:
            words = line.strip().split()
            if len(words) >= 4:
```

```
            del words[3]
        f.write(" ".join(words) + "\n")
```

33. a) read(size) vs read()

- read() (without arguments):

This call reads the entire content of the file or stream from the current position until the end-of-file (EOF) is reached. The entire content is then returned as a single string (in text mode) or bytes object (in binary mode). This is useful when the file size is manageable and you need to process the whole content at once.

- read(size) (with an integer argument size):

This call reads a specific number of bytes or characters (determined by size) from the file or stream, starting from the current position. The method will read up to size bytes/characters, or until EOF is reached, whichever comes first.

b)CSV Program

```
import csv
def StoreSoftware():
    with open("Software.csv", "w", newline='') as f:
        writer = csv.writer(f)
        writer.writerow(["SoftID", "Softname", "Type", "Manhours"])
        while True:
            ID = input("ID: ")
            name = input("Name: ")
            typ = input("Type: ")
            hrs = input("Manhours: ")
            if typ.lower() == "web":
                writer.writerow([ID, name, typ, hrs])
            if input("More? y/n: ") != 'y':
                break
def SearchSoft(hours):
    with open("Software.csv", "r") as f:
        reader = csv.DictReader(f)
        found = False
        for row in reader:
            if int(row["Manhours"]) > hours:
                print(row)
                found = True
        if not found:
            print("No records above", hours)
```

OR

a) the seek() function in python takes two parameters : offset : required, the number of characters to move the file handle. from_where : optional, the reference point from which the file handle is moved relatively.

b)

```
import pickle
def AddEbill():
    with open("Electricity.dat", "ab") as f:
        while True:
            rec = {}
            rec['ConsumerID'] = int(input("ConsumerID: "))
            rec['MeterNo'] = input("MeterNo: ")
            rec['BillAmt'] = float(input("BillAmt: "))
            rec['Paystatus'] = input("Paystatus: ")
            pickle.dump(rec, f)
            if rec['BillAmt'] < 1000:
                print("Low Bill:", rec['ConsumerID'])
```

```
        if input("More? y/n: ") != 'y':
            break
def ShowReport():
    with open("Electricity.dat", "rb") as f:
        try:
            while True:
                rec = pickle.load(f)
                if rec['Paystatus'].lower() == "unpaid":
                    print(rec)
        except:
            pass
```

SECTION - E

34. SQL – MobileMaster & MobileStock

i. SELECT Company, Mname, Price FROM MobileMaster ORDER BY MfgDate DESC;

ii. SELECT * FROM MobileMaster WHERE Mname LIKE 'S%';

iii. SELECT Supplier, Quantity FROM MobileStock WHERE MobileID != 'MB003';

-- OR --

SELECT Company FROM MobileMaster WHERE Price BETWEEN 3000 AND 5000;

35. **i)** pickle module
ii) fout=open(“temp.dat”,”wb”) # statement 2
iii) pickle.load(fin)# statement 3
pickle.dump(rec,fout)#statement 4

***