CBSE Additional Practice Question Paper
Class: XII Session: 2023-24
Computer Science (083)
Marking Scheme

Q No.	Answer	Total Marks
1	a. per%marks	1
2	b. list.append(element)	1
3	b. lcomme T	1
4	b. One block of except statement cannot handle multiple exceptions	1
5	c. Statement 3	1
6	d. dump	1
7	d. dict_student.update(dict_marks)	1
8	b. mean()	1
9	c. 13.5	1
10	PPP – Point to Point Protocol VoIP - Voice Over Internet Protocol	1
11	b. LIKE operator	1
12	d. fetchone	1
13	b. r	1
14	a. file_object.seek(offset [, reference_point])	1
15	d. Interlinking of collection of webpages is called Internet.	1
16	c. TelNet	1
17	a. Both A and R are true and R is the correct explanation for A	1
18	c. A is True but R is False	1
19	Advantages: 1) A dedicated communication channel increases the quality of communication. 2) Suitable for long continuous communication. Disadvantages: 1) Resources are not utilized fully. 2) The time required to establish the physical link between the two stations is too long. *½ mark for each advantage and disadvantage* **OR** **Web browser** Purpose: Receives and displays web content.	2

	Function: Initiates requests to web servers, and receives and displays content for users. **Web server** Purpose: Delivers web content to clients. Function: Listens to incoming requests, processes them, and sends requested content to the client. **Name of Web browsers:** Google Chrome, Mozilla Firefox *1 mark for any one correct difference and 1/2 mark for each two correct examples*	
20	`num1, num2 = 10, 45` `while num1 % num2 == 0:` `____num1+= 20` `____num2+= 30` `else:` `    print('hello')` *½ mark for while* *½ mark for :* *½ mark for correct indentation (inside the block of while)* *½ mark for else*	2
21	`def dispBook(BOOKS):` `    for key in BOOKS:` `        if BOOKS[key][0] not in "AEIOUaeiou":` `            print(BOOKS[key].upper())` `BOOKS = {1:"Python",2:"Internet Fundamentals ",3:"Networking ",4:"Oracle sets",5:"Understanding HTML"}` `dispBook(BOOKS)` *½ mark for for loop* *1 mark for if condition* *½ mark for display in upper case* **OR** `def FindWord(STRING,SEARCH):` `    return (STRING . count (SEARCH) )` `str = input('Enter String : ')` `word = input('Enter word to search : ')` `print('The word', word, 'occurs', FindWord(str,word), 'times')` *½ mark for input* *½ mark for print statement* *1 mark for counting the word and returning the value*	2
22	9$14$19$5$ *½ mark for 9$* *½ mark for 14$* *½ mark for 19$* *½ mark for 5$*	2
23	i. del D['Mumbai'] *1 mark for correct answer*	2

	ii. print(S.split()) *1 mark for correct answer* **OR** my_str = "Computer Science" alternate_chars = my_str[::2] print(alternate_chars) *1.5 mark for logic of alternate characters* *½ mark for printing alternate characters*	
24	% (Percentage): • Matches any sequence of characters (including empty sequence). • Example: LIKE 'T%' matches all those strings starting with the letter 'T'. The string with just 1 character 'T' will also be considered. _ (Underscore): • Matches a single character. • Example: LIKE '_ _T' on the other hand will search for a three letter string, whose 3rd letter is 'T'. At first two places any two character can appear. *1 mark for one correct difference. 1/2 mark each for correct example of each.* **OR** DROP is a DDL command in SQL and can be used to remove tables (or database). Example: 'DROP TABLE STUDENT;' will remove the table STUDENT from the database. DELETE is a DML command used to remove or delete rows/records from a table. Example: 'DELETE FROM STUDENT WHERE PER < 33;' will remove all those records from the table STUDENT where the percentage is less than 33. *1 mark for one correct difference. 1/2 mark each for correct example of each.*	2
25	• COUNT(*) returns the count of all rows in the table, whereas COUNT() is used with Column_Name passed as an argument and counts the number of non-NULL values in a column that is given as an argument. Hence the result may differ. • The SQL command with COUNT(*) may have higher value as it count all rows in the table. *1 mark for suitable reason* *1 mark for mentioning correct command*	2
26	(a) CODE | BNAME | TYPE | MNO | MNAME | ISSUEDATE L102 | Easy Python | Programming | M101 | SNEH SINHA | 2022-10-13 F102 | Untold Story | Fiction | M103 | SARTHAK | 2021-02-23 C101 | Juman Ji | Thriller | M102 | SARA KHAN | 2022-06-12 *1 mark for correct answer*	3

(b)

(i)

NAME	PROJECT
Satyansh	P04
Ranjan	P01
Muneera	P01
Alex	P02
Akhtar	P04

½ mark for correct output

(ii)

NAME	SALARY
Akhtar	125000
Alex	75000

½ mark for correct output

(iii)

NAME	DOJ
Ranjan	2015-01-21
Akhtar	2015-02-01
Muneera	2018-08-19

½ mark for correct output

(iv)

Eid	Name	DOB	DOJ	Salary	Project
E01	Rannja	1990-07-12	2015-01-21	150000	P01
E03	Muneera	1996-11-15	2018-08-19	135000	P01

½ mark for correct output

27 (a) 3

(i)

FID	MIN(FEES)	MAX(FEES)
F01	12000	40000
F04	15000	17000
F03	8000	8000
F05	NULL	NULL

½ mark for correct answer

(ii)

AVG(SALARY)
29500

½ mark for correct answer

(iii)

FNAME	CNAME
Neha	Python
Neha	Computer Network

½ mark for correct answer

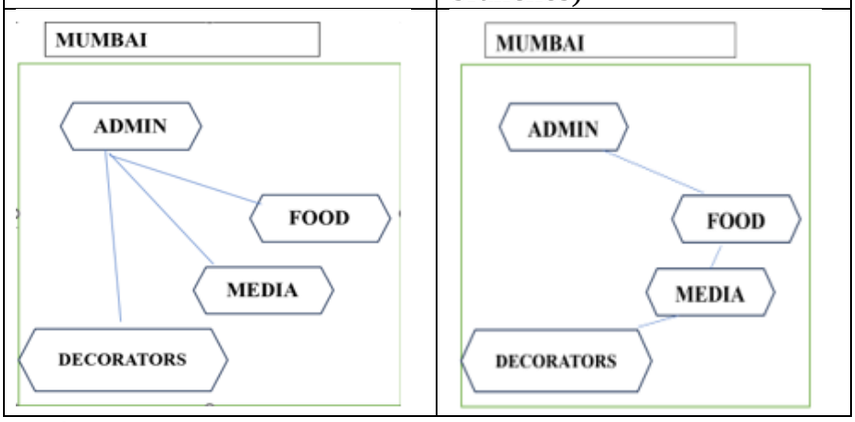
(iv)

FNAME	CNAME	FEES
Anishma	Grid Computing	40000
Neha	Python	17000

½ mark for correct answer

	(b) DESC or DESCRIBE command *1 mark for correct answer*	
28	``` def Count(): F=open('Gratitude.txt') T=F.readlines() X=1 for i in T: print('Line',X,':',i.count('e')) X=X+1 F.close() Count() ``` *½ mark for function header* *½ mark for opening and closing the file* *½ mark for reading lines* *½ mark for loop* *½ mark for count function/or any other alternate correct statement(s)* *½ mark for counter* **OR** ``` def Start_with_I(): F=open('Gratitude.txt') T=F.readlines() for i in T: if i[0] in 'Ii': print(i,end='') F.close() Start_with_I() ``` *½ mark for function header* *½ mark for opening and closing the file* *½ mark for reading lines* *½ mark for loop* *½ mark for if condition* *½ mark for print statment*	3
29	(i) Candidate Keys : ADMNO, ROLLNO *1 mark for correctly writing both names of candidate keys. OR ½ mark for specifying any one candidate key correctly.* (ii) Degree-8, Cardinality=4 *½ mark for degree and ½ mark for cardinality* (iii) Update result set SEM2=SEM2+.03*SEM2 where SEM2 between 70 and 100; *½ mark for writing Update result set part correctly* *½ mark for writing SEM2=SEM2+.03*SEM2 where SEM2 between 70 and 100; correctly.*	3
30	``` Stu_dict={5:(87,68,89), 10:(57,54,61), 12:(71,67,90), 14:(66,81,80), 18:(80,48,91)} ```	3

	`` Stu_Stk=[] def Push_elements(Stu_Stk, Stu_dict): for Stu_ID, marks in Stu_dict.items(): if marks[2]>=80: Stu_Stk.append(Stu_ID) def Pop_elements(Stu_Stk): while len(Stu_Stk)>0: print(Stu_Stk.pop()) if not Stu_Stk: print('Stack Empty') Push_elements(Stu_Stk, Stu_dict) Pop_elements(Stu_Stk) ``` *1.5 marks for correct implementation of Push_elements()* *1.5 marks for correct implementation of Pop_elements()*	
31	``` import csv def maxsalary(): f=open('record.csv', 'r') reader=csv.reader(f) skip_header = True max= 0 for row in reader: if skip_header: skip_header = False else: if(int(row[3])>max): max=int(row[3]) rec=row print('Row with the highest salary : ', rec) f.close() maxsalary() ``` *½ mark for importing module* *½ mark for function definition* *½ mark for opening and closing file* *½ for reader object* *½ for skipping first row (i.e. header)* *1 mark for calculating maximum salary* *½ mark for displaying record having maximum salary*	4
32	``` import pickle def expensiveProducts(): with open('INVENTORY.DAT', 'rb') as file: expensive_count = 0 while True: try: product_data = pickle.load(file) product_id, product_name, quantity, price = product_data if price > 1000: print("Product ID:", product_id) expensive_count += 1 except EOFError: break print("Total expensive products: ", expensive_count) expensiveProducts() ``` *½ mark for function definition*	4

	½ mark for opening and closing file *½ mark for correct try and except block* *1.5 mark identifying and displaying details of expensive products* *1 mark for displaying count of expensive products*	
33	i. The most appropriate location of the server inside the MUMBAI campus is ADMIN building due to the maximum number of computers in it. *½ mark for mentioning the branch and ½ mark for proper justification* ii. Cable Layout Star Topology (Based on server location) / Bus Topology (Based on minimum distance between branches)  *1 mark for drawing any valid cable layout* iii. Switch or Hub *1mark for suggesting the correct device* iv. c. Video Conferencing *1 mark for correct answer* v. (a) WAN (b) LAN *½ mark for mentioning WAN and ½ mark for mentioning LAN*	5
34	i.	5

	seek()	tell()
Purpose	Repositions the file pointer to a specific location within a file	Returns the current position of the file pointer
Syntax	seek(offset [,reference point])	tell()
Parameters	Requires specifying the offset and an optional reference point	Requires no parameters

2 marks for mentioning two correct differences.
OR
1 marks for mentioning only one correct differences.

ii.

```
import pickle
def  COPY_REC():
    In_file = open('FLIGHT.DAT','rb')
    out_file = open('RECORD.DAT','wb')
    try:
```

```
        while True:
            data = pickle.load(In_file)
            if data[3] == 'DELHI' and data[4] == 'MUMBAI':
                pickle.dump(data,out_file)
    except:
        In_file.close()
        out_file.close()

COPY_REC()
```

½ mark for function definion
½ mark for correctly opening and closing file
½ mark for correct try and except block
1.5 marks for writing required data in RECORD.DAT

OR

i.

Binary	CSV
1. pickle module to be used	1. csv module is used
2. Data is stored in binary format(0s and 1s) and is not in human readable form using any plain text editor.	2. Data is stored in tabular fashion and comma separated by default. The file can be read by any spreadsheet software or text editor.
3. File extension .dat/.pdf/.exe etc.	3. File extension .csv

2 marks for mentioning two correct differences.
OR
1 marks for mentioning only one correct differences.

ii.

```
import pickle

def findBook(price):
    with open('BOOK.DAT', 'rb') as file:
        while True:
            try:
                book_record = pickle.load(file)
                for item in book_record:
                    book_price = book_record[item][2]
                    if book_price >= price:
                        print(item, book_record[item])
            except EOFError:
                break

findBook(50)
```

½ mark for function definion
½ mark for correctly opening and closing file
½ mark for correct try and except block
1.5 marks for displying required records

35	(i)	5

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table.
Constraints –
NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

½ mark for correct definition, ½ mark for correct example (anyone)

(ii)

a) password='tiger'
b) mycursor = con1.cursor()
c) query = 'delete from ITEM where Iname = "{}" '.format.(item_name)
d) con1.commit()

1 mark for each correct statement

OR

(i)

Candidate Key: A candidate key is a set of attributes in a relation that can uniquely identify each tuple (row). A relation can have multiple candidate keys, but only one of them is chosen as the primary key.
Alternate Key: An alternate key is a candidate key that is not selected as the primary key.

1 mark for any one correct difference.

(ii)

a) import mysql.connector as mysql
b) mycursor = con1.cursor()
c) query = 'SELECT * FROM ITEM where Price > {}'.format(5000)
d) data = mycursor.fetchall()

1 mark for each correct statement