# COMPUTER SCIENCE

## Paper – 2

## (PRACTICAL)

***(Maximum Marks: 30)***

***(Time allowed: Three hours)***

**(***Candidates are allowed additional 15 minutes for only reading the paper.*

*They must NOT start writing during this time.***)**

---

***The total time to be spent on the Planning Session and the Examination Session is Three hours.***

*Planning session: 90 minutes*

*Examination session : 90 minutes*

**Note: Candidates are to be permitted to proceed to the Examination Session only after 90 minutes of the Planning session are over.**

---

*This paper consists of **three** problems from which candidates are required to attempt **any one** problem.*

Candidates are expected to do the following:

1. Write an algorithm for the selected problem. **[3]**
   (Algorithm should be expressed clearly using any standard scheme such as pseudo code or in steps which are simple enough to be obviously computable.)

2. Write a program in **JAVA** language. The program should follow the algorithm and should be logically and syntactically correct. **[5]**

3. Document the program using mnemonic names / comments, identifying and clearly describing the choice of data types and meaning of variables. **[2]**

4. Code / Type the program on the computer and get a printout ( hard copy ). Typically, this should be a program that compiles and runs correctly. **[2]**

5. Test run the program on the computer using the given sample data and get a printout of the output in the format specified in the problem. **[5]**

6. Viva-Voce on the **Selected Problem**. **[3]**

In addition to the above, the practical file of the candidate containing the practical work related to programming assignments done during the year is to be evaluated as follows:

- Programming assignments done throughout the year (by the teacher) **[5]**
- Programming assignments done throughout the year (by the Visiting Examiner) **[5]**

---

**This Paper consists of 5 printed pages and 1 blank page.**

*Solve **any one** of the following Problems.*

**Question 1**

A company manufactures packing cartons in four sizes, i.e. cartons to accommodate 6 boxes, 12 boxes, 24 boxes and 48 boxes. Design a program to accept the number of boxes to be packed (N) by the user (maximum up to 1000 boxes) and display the break-up of the cartons used in descending order of capacity (i.e. preference should be given to the highest capacity available, and if boxes left are less than 6, an extra carton of capacity 6 should be used.)

Test your program with the following data and some random data:

**Example 1**

**INPUT**: N = 726

**OUTPUT**:

| | |
|---|---|
| | 48 × 15 = 720 |
| | 6 × 1 = 6 |
| Remaining boxes | = 0 |
| Total number of boxes | = 726 |
| Total number of cartons | = 16 |

**Example 2**

**INPUT**: N = 140

**OUTPUT**:

| | |
|---|---|
| | 48 × 2 = 96 |
| | 24 × 1 = 24 |
| | 12 × 1 = 12 |
| | 6 × 1 = 6 |
| Remaining boxes | 2 × 1 = 2 |
| Total number of boxes | = 140 |
| Total number of cartons | = 6 |

**Example 3**

**INPUT**: N = 4296

**OUTPUT**: INVALID INPUT

**Question 2**

The result of a quiz competition is to be prepared as follows:

The quiz has five questions with four multiple choices (A, B, C, D), with each question carrying 1 mark for the correct answer. Design a program to accept the number of participants N such that N must be greater than 3 and less than 11. Create a double dimensional array of size (N×5) to store the answers of each participant row-wise. Calculate the marks for each participant by matching the correct answer stored in a single dimensional array of size 5. Display the scores for each participant and also the participant(s) having the highest score.

**Example**: If the value of N = 4, then the array would be :

| | Q.1 | Q.2 | Q.3 | Q.4 | Q.5 |
|---|---|---|---|---|---|
| Participant 1 | A | B | B | C | A |
| Participant 2 | D | A | D | C | B |
| Participant 3 | A | A | B | A | C |
| Participant 4 | D | C | C | A | B |

| Key to the question: | D | C | C | A | B |
|---|---|---|---|---|---|

**Note: Array entries are line fed (i.e. one entry per line)**

Test your program for the following data and some random data:

**Example 1**

**INPUT**: N = 5
Participant 1 D A B C C
Participant 2 A A D C B
Participant 3 B A C D B
Participant 4 D A D C B
Participant 5 B C A D D
**Key :** B C D A A

**OUTPUT**: **Scores:**
Participant 1 = 0
Participant 2 = 1
Participant 3 = 1
Participant 4 = 1
Participant 5 = 2
Highest score: Participant 5

**Example 2**

**INPUT**: N = 4
Participant 1 A C C B D
Participant 2 B C A A C
Participant 3 B C B A A
Participant 4 C C D D B
**Key :** A C D B B

**OUTPUT**: **Scores:**
Participant 1 = 3
Participant 2 = 1
Participant 3 = 1
Participant 4 = 3
Highest score: Participant 1
Participant 4

**Example 3**

**INPUT**: N = 12

**OUTPUT:** INPUT SIZE OUT OF RANGE.

**Question 3**

**Caesar Cipher** is an encryption technique which is implemented as ROT13 ('rotate by 13 places'). It is a simple letter substitution cipher that replaces a letter with the letter 13 places after it in the alphabets, with the other characters remaining unchanged.

**ROT13**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A/a | B/b | C/c | D/d | E/e | F/f | G/g | H/h | I/i | J/j | K/k | L/l | M/m |
| ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ |
| N/n | O/o | P/p | Q/q | R/r | S/s | T/t | U/u | V/v | W/w | X/x | Y/y | Z/z |

Write a program to accept a plain text of length L, where L must be greater than 3 and less than 100.

Encrypt the text if valid as per the Caesar Cipher.

Test your program with the sample data and some random data:

**Example 1**

**INPUT**: Hello! How are you?

**OUTPUT**: The cipher text is:
Uryyb? Ubj ner lbh?

**Example 2**

**INPUT**: Encryption helps to secure data.

**OUTPUT**: The cipher text is:
Rapelcgvba Urycf gb frpher gngn.

**Example 3**

**INPUT**: You
**OUTPUT**: INVALID LENGTH