# COMPUTER SCIENCE

**Paper – 2**

## (PRACTICAL)

**(Three hours)**

**Maximum Marks: 30**

*(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 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 <u>only after</u> the 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.
   (Algorithm should be expressed clearly using any standard scheme such as pseudo code or in steps which are simple enough to be obviously computable) **[3]**

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 print out ( 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 print out of the output in the format specified in the problem. **[5]**

6. Viva-Voce on the **<u>Selected Problem</u>**. **[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]**

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

**Question 1**

An Evil number is a positive whole number which has even number of 1's in its binary equivalent.

Example: Binary equivalent of 9 is 1001, which contains even number of 1's.

A few evil numbers are 3, 5, 6, 9....

Design a program to accept a positive whole number and find the binary equivalent of the number and count the number of 1's in it and display whether it is a Evil number or not with an appropriate message.

Output the result in format given below:

**Example 1**

INPUT : 15

BINARY EQUIVALENT : 1111

NO. OF 1's : 4

OUTPUT : EVIL NUMBER

**Example 2**

INPUT : 26

BINARY EQUIVALENT : 11010

NO. OF 1's : 3

OUTPUT : NOT AN EVIL NUMBER

**Question 2**

The encryption of alphabets are to be done as follows:

A = 1
B = 2
C = 3
.
.
.
Z = 26

The potential of a word is found by adding the encrypted value of the alphabets.

Example: KITE
Potential = 11 + 9 + 20 + 5 = 45

Accept a sentence which is terminated by either " . " , " ? " or " ! ". Each word of sentence is separated by single space. Decode the words according to their potential and arrange them in ascending order.

Output the result in format given below:

**Example 1**

INPUT : THE SKY IS THE LIMIT.

POTENTIAL :
THE = 33
SKY = 55
IS = 28
THE = 33
LIMIT = 63

OUTPUT : IS THE THE SKY LIMIT

**Example 2**

INPUT : LOOK BEFORE YOU LEAP.

POTENTIAL :
LOOK = 53
BEFORE = 51
YOU = 61
LEAP = 34

OUTPUT : LEAP BEFORE LOOK YOU

**Question 3**

Given a square matrix M [ ] [ ] of order 'n'. The maximum value possible for 'n' is 10. Accept three different characters from the keyboard and fill the array according to the instruction given below.

Fill the upper and lower elements formed by the intersection of the diagonals by character 1.

Fill the left and right elements formed by the intersection of the diagonals by character 2.

Fill both the diagonals by character 3.

Output the result in format given below:

**Example 1**

ENTER SIZE : 4

INPUT : FIRST CHARACTER '*'
SECOND CHARACTER '?'
THIRD CHARACTER '#'

OUTPUT :

```
# * * #
? # # ?
? # # ?
# * * #
```

**Example 2**

ENTER SIZE : 5

INPUT : FIRST CHARACTER '$'
SECOND CHARACTER '!'
THIRD CHARACTER '@'

OUTPUT :

```
@ $ $ $ @
! @ $ @ !
! ! @ ! !
! @ $ @ !
@ $ $ $ @
```

**Example 3**

ENTER SIZE : 65

OUTPUT : SIZE OUT OF RANGE