- Use C syntax only.
- Avoid using global variable.
- Do check for runtime error conditions.
- Meaningful variable naming convention / comments, proper modularity and writing style will fetch more marks.

1. You have a softcopy of a story book stored in a file named MyStory.txt. Write a C program to find the word(s) that appeared least (non-zero) in this book.
2. Consider the following variable declarations:

   ```
   int *ipMatIP1, *ipMatIP2, iRow1, iCol1, iRow2, iCol2;
   ```

   Assume two matrices are stored in the first two variables with iRow1, iCol1, and iRow2, iCol2 as their respective dimensions. Design a matrix multiplication C function and write the code.
3. Write a complete well documented C program that accepts an integer from the command line and prints the prime factorization on screen and a file with filename as the input integer and extension as .txt.
   eg. If input number is 123, the file name should be 123.txt
   For any invalid input, it should be able to print error message and quit.
4. Write a program that accepts only single alphabetical characters. On encountering an non-alphabet it terminates after printing all the alphabets entered so far in shorted order.
5. Write a program to print a histogram of the frequencies of different characters in its input.
6. Write a program to "fold" long input lines into two or more shorter lines after the last non-blank character that occurs before the n-th column of input. Make sure your program does something intelligent with very long lines, and if there are no blanks or tabs before the specified column.
7. Write a program to check a C program for rudimentary syntax errors like unbalanced parentheses, brackets and braces. Don't forget about quotes, both single and double, escape sequences and comments.

You may share the marking scheme (even to the students) for better understanding and uniformity of marking across colleges.
Marking scheme:

1. No marks for incomplete code or code with compilation error (code that does not run).
2. Code that runs and takes input correctly - 30% marks
3. Code that is able to print the largest numbers (max four) - 30% marks
4. Program designed with appropriate functions - 15% marks
5. Proper error handling - 10% marks
6. No global Variable - 5% marks
7. Meaningful variable naming convention, appropriate comments, proper alignment - 10% marks

Testing Codes:

Check with different valid input sequence - ascending / descending / random order with (or without) repetition. Also test with invalid inputs.

Marking Table:

RollNo	Input (30)	Output (30)	Design (15)	Error (10)	No Global Variable (05)	Readability (10)	Total (100)