Computing Lab Assignments

Lab 2

ISI, Kolkata

August 5, 2025

Outline

Outline

RLE

Run-length encoding (RLE) is a form of lossless data compression in which runs of data (consecutive occurrences of the same data value) are stored as a single occurrence of that data value and a count of its consecutive occurrences, rather than as the original run.

Example: Run-Length Encoded form of *aaabccd* is *a*3*bc*2*d*.

Tasks:

- Write a program to convert a given string s to s', its run-length encoded form. This means that s will contain the same sequence of distinct characters as s, but any $m > 1$ consecutive occurrences of a character will be replaced by a single occurrence of the character immediately followed by the integer m (in base 10).
  The string s should be read from the terminal. It may contain letters, digits, blanks and tabs (\t), but no newline. The length of the string s will not be known to you in advance.

RLE

Tasks:

- Add code to your program so that it prints the character that occurs consecutively the maximum number of times. For the example above, your program should print *a*. If the maximum number of consecutive occurrences is the same for two or more characters, you may print any one.
- Modify your program so that it decodes a given run-length encoded string *s* to its original form *s*. For example, given *a3bc2d*, your program should print *aaabccd*. Note that, given *abcd*, your program should print *abcd* itself. The input format will be the same as that for encoding.

Outline

Letter Frequency Counter

Goal: Read input from the terminal and count the number of occurrences of each letter `a-z`.

Example 1:

- Input: `C Programming is FUN!!`
- Output:

Letter	Count	Letter	Count	Letter	Count
a	1	i	2	r	2
c	1	m	2	s	1
f	1	n	2	u	1
g	2	o	1	p	1

Letter Frequency Counter

Constraints:

- Input may span **multiple lines**.
- Input may contain **digits**, **punctuation**, **spaces**, **tabs**, and other **non-letter** characters. **Ignore these characters!!**
- Do **not distinguish** between uppercase and lowercase letters.

Task: Write a `C` program that:

- Reads characters from standard input,
- Counts how many times each letter `a-z` appears, with above constraints,
- Prints the frequency of each letter (in alphabetical order).

Outline

String Reversal

Goal: Take a word with no spaces as input and print it in reverse.

Input	Output
`computer`	`retupmoc`
`OpenAI`	`IAnepO`
`Beamer`	`remaeB`

Table: Examples

Constraints:

- Input has **no whitespace (blanks / tabs)**
- At most **80 characters**
- Input has no **newlines**

Task: Write a `C` program that:

- Reads a string as input
- Reverses the string
- Prints the reversed string to standard output.

Outline

Multivariate Polynomial Representation

A multivariate polynomial can be represented as a sum of non-zero terms. We store:

- The number of terms
- A list of non-zero terms, each with:
  - A coefficient
  - Exponents for each variable

Example: 4-variable polynomial with variables a, b, c, d

$$5a^2bd^3 - 2d^5 + abcd + 31$$

Explanation:

- This polynomial has 4 non-zero terms.
- Each term stores: coefficient + exponents for a, b, c, d

Multivariate Polynomial Representation

Example Polynomial:

$$5a^2bd^3 - 2d^5 + abcd + 31$$

Term Representation:

Number of terms = 4

Term	Coeff	a	b	c	d
Term 0	5	2	1	0	3
Term 1	-2	0	0	0	5
Term 2	1	1	1	1	1
Term 3	31	0	0	0	0

Storage Structure:

```
1 #define NVAR 5
2 #define MAX_TERM 1000
3
4 typedef struct {
5     int nterm;
6     int term[MAX_TERM][1 + NVAR
      ];
7 } mvpoly;
```

Multivariate Polynomial Representation

Print Function:

```
void polyprint (mvpoly f) {
    int i, k;
    for (i = 0; i < f.nterm; ++i) {
        if ((i > 0) && (f.term[i][0] >= 0))
            printf("+");
        if (f.term[i][0] != 1)
            printf("%d", f.term[i][0]);
        for (k = 1; k <= NVAR; ++k) {
            if (f.term[i][k])
                printf("%c", 'a' + (char)(k - 1));
            if (f.term[i][k] > 1)
                printf("^%d", f.term[i][k]);
        }
    }
}
```

Multivariate Polynomial Product

Task 1: Write a function to compute the product of two multivariate polynomials.

Function prototype:

```
mvpoly polyprod(mvpoly f, mvpoly g);
```

Important note: When multiplying two multivariate polynomials, the same product monomial may appear multiple times during the monomial-by-monomial multiplication. To correctly represent the polynomial, *like terms must be combined* (collected) in the output.

Example: $(a + b) \times (a + 2b)$

$$\text{Correct output:} \quad a^2 + 3ab + 2b^2$$

$$\text{Incorrect output:} \quad a^2 + 2ab + ab + 2b^2$$

Multivariate Polynomial Product

Task 2: The `main()` function

Use $n = 5$ variables a, b, c, d, e. Compute and print the polynomials $(a + b + c + d + e)^i$ for $i = 1, 2, 3, 4, 5$.

Sample output for $n = 3$:

$$
\begin{aligned}
(a+b+c)^1 &= a + b + c \\
(a+b+c)^2 &= a^2 + 2ab + 2ac + b^2 + 2bc + c^2 \\
(a+b+c)^3 &= a^3 + 3a^2b + 3a^2c + 3ab^2 + 6abc + 3ac^2 + b^3 + \\
&\qquad 3b^2c + 3bc^2 + c^3
\end{aligned}
$$