String Problem Statements

1. Log File Anomaly Detection

Question: Parse a log file to identify and count specific error codes. Calculate the frequency of each error type and flag any that exceed a predefined threshold. The log entries are not uniform, requiring pattern matching with regular expressions.

Sample Input:

```
[2025-08-15 10:01:23] INFO: User 'Alice' logged in.
[2025-08-15 10:02:15] ERROR_DB: Connection failed.
[2025-08-15 10:03:01] WARNING: High memory usage detected.
[2025-08-15 10:04:45] ERROR_DB: Query timeout.
[2025-08-15 10:05:00] INFO: Service restart successful.
[2025-08-15 10:05:22] ERROR_NET: Server unreachable.
[2025-08-15 10:06:10] ERROR_DB: Disk full.
```

Sample Output:

```
Error Frequencies:
ERROR_DB: 3
ERROR_NET: 1
Threshold Exceeded: ERROR_DB
```

---

2. DNA Sequence Alignment

Question: Given two DNA sequences, implement a scoring system to find the optimal alignment. This involves string manipulation, calculating a similarity score, and handling mismatches and gaps.

Sample Input:

```
Sequence 1: ATCGATTAC
Sequence 2: AATCGATAC
```

Sample Output:

```
Alignment:
A T C G A T T A C
A A T C G A - T A C
Score: 4 (Mismatch Penalty: -1, Gap Penalty: -2)
```

3. Secure Password Strength Checker

Question: Develop a function that analyzes a password string for strength. The analysis includes checking for a minimum length, the presence of uppercase letters, lowercase letters, numbers, and special characters. It must also calculate an entropy score based on the character set size and password length.

Sample Input:

```
Password: MyStrongP@ssw0rd!
```

Sample Output:

```
Strength: Strong
Length: 17 (OK)
Lowercase: Yes (OK)
Uppercase: Yes (OK)
Numbers: Yes (OK)
Special Chars: Yes (OK)
Entropy Score: 110.1 bits
```

---

4. URL Shortener Service

Question: Given a long URL, generate a unique, short string identifier. This requires a hashing algorithm and collision detection to ensure no two URLs get the same short code.

Sample Input:

```
URL: https://www.example.com/very/long/path/to/a/specific/resource?
id=123456789&user=test
```

Sample Output:

```
Short URL: https://example.com/aBcDeFg
```

---

5. Text Compression Algorithm

Question: Implement a basic run-length encoding (RLE) algorithm. This involves iterating through a string, identifying consecutive repeating characters, and replacing them with a count and the character.

Sample Input:

```
Input: AAABBCDDDDDA
```

Sample Output:

```
Compressed: 3A2B5D1A
```

---

6. Automated Subtitle Synchronization

Question: Given a subtitle file with timecodes (e.g., "00:01:23,456 --> 00:01:25,678"), and a string of dialogue, shift all timecodes forward or backward by a specific duration. This requires parsing time strings, performing time arithmetic, and reformatting the output string.

Sample Input:

```
Original Subtitle: 00:01:23,456 --> 00:01:25,678
Dialogue: Hello world.
Shift: +2.5 seconds
```

Sample Output:

```
New Subtitle: 00:01:25,956 --> 00:01:28,178
Dialogue: Hello world.
```

---

7. Financial Transaction Parser

Question: Parse a string representing a financial transaction. Extract the transaction type, amount, account number, and date, then perform calculations like summing all deposits or withdrawals.

Sample Input:

```
Transaction String: DEPOSIT: $1,234.56, Acct: 987654321, Date: 2025-08-15
```

Sample Output:

```
Type: DEPOSIT
Amount: 1234.56
Account: 987654321
Date: 2025-08-15
```

8. Source Code Comment Extractor

Question: Write a program that reads a C++ source file as a single string and extracts all single-line and multi-line comments. This requires handling different comment syntaxes and nested structures.

Sample Input:

```
// This is a single-line comment.
int main() {
  /*
    This is a multi-line comment.
    It spans several lines.
  */
  int x = 10; // Inline comment.
  /* Another multi-line comment */
  return 0;
}
```

Sample Output:

```
Extracted Comments:
This is a single-line comment.
This is a multi-line comment.
It spans several lines.
Inline comment.
Another multi-line comment
```

---

9. Data Validation from a CSV

Question: Given a line of text from a CSV file, parse the string by comma delimiters. Validate that each field is of the correct data type (e.g., age is an integer, phone number matches a specific pattern) and calculate an error score for the row.

Sample Input:

```
CSV Line: John Doe,42,123 Main St,555-1234
```

Sample Output:

```
Fields:
Name: John Doe (Valid)
Age: 42 (Valid)
Address: 123 Main St (Valid)
```

```
Phone: 555-1234 (Valid)
Error Score: 0
```

---

10. Image Metadata Editor

Question: Given a string representing image metadata in a format like EXIF, parse it to find specific tags (e.g., "Make," "Model," "DateTimeOriginal"). Modify the values of these tags and output the new metadata string.

Sample Input:

```
Metadata: [Make: Nikon], [Model: D750], [DateTimeOriginal: 2025:08:15 10:30:00]
Tag to change: DateTimeOriginal
New Value: 2025:08:16 12:00:00
```

Sample Output:

```
New Metadata: [Make: Nikon], [Model: D750], [DateTimeOriginal: 2025:08:16
12:00:00]
```

---

11. Network Packet Payload Inspector

Question: A developer receives a hexadecimal string representing a network packet payload. The task is to parse this string, convert parts of it to different data types (e.g., IP addresses, port numbers, message body) and then perform a checksum calculation to verify data integrity.

Sample Input:

```
Payload: 45000034a780000040062f837f0000017f0000011f9000501f2f0000416800000000...
```

Sample Output:

```
Source IP: 127.0.0.1
Destination IP: 127.0.0.1
Source Port: 8080
Destination Port: 80
Checksum: 0x2f83
Data Integrity: OK
```

---

12. Natural Language Processing (NLP) - Word Counter

Question: Count the frequency of each word in a given text string. This requires tokenization (splitting the string into words), handling punctuation, converting to a consistent case (lowercase), and storing the counts in a map.

Sample Input:

```
Text: "The quick brown fox jumps over the lazy dog. The fox is quick."
```

Sample Output:

```
Word Counts:
the: 3
quick: 2
fox: 2
brown: 1
jumps: 1
over: 1
lazy: 1
dog: 1
is: 1
```

---

13. Scientific Notation Converter

Question: Parse a string representing a number in scientific notation and convert it to a standard decimal number string, handling both positive and negative exponents and ensuring correct precision.

Sample Input:

```
Input: 6.022e+23
```

Sample Output:

```
Decimal: 602,200,000,000,000,000,000,000
```

---

14. JSON Object Manipulator

Question: Given a string representing a simple JSON object, find and modify a specific key's value. This involves parsing the string to find the key, updating the corresponding value, and then reconstructing the JSON string.

Sample Input:

```
{ "name": "John Doe", "age": 30, "isStudent": false }
Key to change: age
New Value: 35
```

Sample Output:

```
{ "name": "John Doe", "age": 35, "isStudent": false }
```

---

15. User-Agent String Parser

Question: A web server receives a user-agent string from a browser. Extract and identify the browser name, operating system, and version number.

Sample Input:

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/108.0.0.0 Safari/537.36
```

Sample Output:

```
OS: Windows NT 10.0
Browser: Chrome
Version: 108.0.0.0
```

---

16. Inventory Management System

Question: Each item in an inventory is represented by a unique string ID with a specific format (e.g., "SKU-PROD-2025-08-15-001"). Parse this string to extract the product code, manufacturing date, and a sequential number. Then, generate the next valid string ID.

Sample Input:

```
Current ID: SKU-PROD-2025-08-15-001
```

Sample Output:

```
Product Code: PROD
Date: 2025-08-15
Sequence: 001
Next ID: SKU-PROD-2025-08-15-002
```

---

17. Simple Chatbot

Question: Given a user input string, the chatbot must identify keywords to determine the user's intent. Based on the identified keywords, it formulates a dynamic response string.

Sample Input:

```
User Input: "What is the weather like today?"
```

Sample Output:

```
Chatbot Response: "I'm sorry, I cannot provide real-time weather information.
Please check a weather app."
```

---

18. Cipher Decryption

Question: Implement a decryption function for a simple substitution cipher. Given an encrypted message string and a key string (which defines the character mapping), decrypt the message.

Sample Input:

```
Encrypted Message: "Ebiil Tloia"
Key: abcdefghijklmnopqrstuvwxyz -> zyxwvutsrqponmlkjihgfedcba
```

Sample Output:

```
Decrypted Message: "Hello World"
```

---

19. Command-Line Argument Parser

Question: Write a function that takes a command-line string and parses it to extract the program name and a map of flags and their corresponding values.

Sample Input:

```
Command: ./myprogram --file=data.txt --verbose -o results.csv --user=admin
```

Sample Output:

```
Program Name: myprogram
Arguments:
file: data.txt
verbose: (flag)
o: results.csv
user: admin
```

---

20. Book Index Generator

Question: Given a text string representing a book chapter, identify all unique words that are not common 'stop words' (e.g., "the," "a," "is"). Create an alphabetical index of these words, along with the line number on which they first appear.

Sample Input:

```
Text:
Line 1: The cat sat on the mat.
Line 2: A lazy dog slept nearby.
Line 3: The cat chased the dog.
```

Sample Output:

```
Index:
cat: 1
chased: 3
dog: 2
lazy: 2
mat: 1
nearby: 2
sat: 1
slept: 2
```