LLD machine coding questions

💡 https://chat.openai.com/share/49b6ce9b-1c72-4103-bc71-300d9f94ad9e

Games

Game: Hangman

- One player thinks of a word and the other player tries to guess it by suggesting letters within a certain number of guesses.
- The word to guess is represented by a row of dashes, representing each letter of the word.
- If the guessing player suggests a letter that occurs in the word, the other player fills in the blanks with that letter in the correct positions.
- If the guessing player suggests a letter that does not occur in the word, the other player draws one element of a hangman diagram as a tally mark.
- The game ends when the guessing player guesses the word correctly or completes the hangman diagram.

Requirements:

- Create a command-line application for playing Hangman.

- Initialize the game with a word to guess.
- Display the word as dashes at the beginning.
- Accept letters from the guessing player as input.
- Update the word display with correctly guessed letters.
- Draw a hangman diagram as incorrect guesses are made.
- End the game when the word is guessed correctly or the hangman diagram is completed.

Input Format:

- Letters guessed by the player.

Example:

```
_ _ _ _ _
Guess a letter: A
_ A _ _ _
Guess a letter: E
_ A _ _ E
Guess a letter: T
_ A T _ E
...
```

Game: Number Guessing Game

Rules:

- One player thinks of a number between a predefined range (e.g., 1-100) and the other player tries to guess it.
- After each guess, the other player provides feedback whether the guess is too high, too low, or correct.

Requirements:

- Create a command-line application for playing the Number Guessing Game.

- Initialize the game with a random number within a predefined range.
- Accept guesses from the player.
- Provide feedback on each guess (too high, too low, or correct).
- End the game when the correct number is guessed.

Input Format:

- Integer guesses from the player.

Example:

```
Guess a number between 1 and 100: 50
Too low. Try again.
Guess a number between 1 and 100: 75
Too high. Try again.
Guess a number between 1 and 100: 60
Too low. Try again.
Guess a number between 1 and 100: 65
Correct! The number was 65.
```

Game: Rock, Paper, Scissors

Rules:

- Two players simultaneously choose either rock, paper, or scissors.
- Rock beats scissors, scissors beats paper, and paper beats rock.
- If both players choose the same item, it is a tie and the game is played again.

Requirements:

- Create a command-line application for playing Rock, Paper, Scissors.
- Accept the choices of both players.
- Determine the winner based on the rules.
- Handle ties by playing the game again.

Input Format:

- Choices of both players (rock, paper, or scissors).

Example:

```
Player 1, enter your choice (rock, paper, scissors): rock
Player 2, enter your choice (rock, paper, scissors): scissors
Player 1 wins! Rock beats scissors.
```

Command-line applications

Todo list

Let's design a simple command-line application to manage a todo list. The application should allow users to add tasks, mark tasks as completed, view the list of tasks, and exit the application.

Requirements:

- Create a command-line application for managing a todo list.
- Initialize an empty list of tasks.
- Accept commands from the user to add a task, mark a task as completed, view the list of tasks, and exit the application.
- When adding a task, prompt the user for the task description.
- When marking a task as completed, prompt the user for the task number.
- When viewing the list of tasks, display the task numbers and descriptions.
- Use the word "exit" to quit the application.

Input Format:

- "add" command followed by the task description to add a new task.
- "done" command followed by the task number to mark a task as completed.
- "view" command to display the list of tasks.
- "exit" command to exit the application.

Example:

```
add Buy groceries
add Clean the house
view
done 1
view
exit
```

Calculator

Let's create a command-line application for a simple calculator. The calculator should be able to perform basic arithmetic operations like addition, subtraction, multiplication, and division on two numbers entered by the user.

Requirements:

- Create a command-line application for a basic calculator.
- Accept two numbers and an arithmetic operator from the user.
- Perform the operation and display the result.
- Handle division by zero by displaying an error message.
- Use the word "exit" to quit the application.

Input Format:

- Two numbers separated by a space, followed by the arithmetic operator (+, -, *, /).
- Use "exit" to quit the application.

Example:

```
5 2 +
7 3 *
```

```
10 0 /
exit
```

Simple note-taking tool

Let's design a command-line application for a simple note-taking tool. The application should allow users to create new notes, view existing notes, and delete notes.

Requirements:

- Create a command-line application for managing notes.
- Initialize an empty list of notes.
- Accept commands from the user to add a note, view all notes, delete a note, and exit the application.
- When adding a note, prompt the user for the note content.
- When viewing all notes, display the list of note numbers and contents.
- When deleting a note, prompt the user for the note number to delete.
- Use the word "exit" to quit the application.

Input Format:

- "add" command followed by the note content to add a new note.
- "view" command to display all notes.
- "delete" command followed by the note number to delete a note.
- "exit" command to exit the application.

Example:

```

add Buy milk
add Call John
view
delete 1

```

```
view
exit
```

Questions

1. **Calculator Application:** Create a command-line calculator that can perform basic arithmetic operations (addition, subtraction, multiplication, division) on two numbers.
2. **Hangman Game:** Implement a command-line version of the Hangman game. The computer selects a word, and the player must guess the word letter by letter.
3. **Number Guessing Game:** Develop a number guessing game where the computer randomly selects a number between a specified range, and the player must guess the number.
4. **Rock, Paper, Scissors Game:** Create a command-line version of the Rock, Paper, Scissors game where two players can play against each other.
5. **Quiz Application:** Build a quiz application that asks the user multiple-choice questions and provides feedback on the answers.
6. **Todo List Application:** Develop a command-line todo list application that allows users to add, remove, and list tasks.
7. **Word Counter:** Create a program that takes a text input and counts the number of words, characters, and lines in the text.
8. **Simple Chatbot:** Implement a simple chatbot that can respond to basic questions or engage in simple conversation with the user.
9. **File Manager:** Build a command-line file manager that allows users to navigate, create, delete, and edit files and directories.
10. **Calendar Application:** Develop a command-line calendar application that allows users to view and manage their schedule, add events, and set reminders.
11. **Palindrome Checker:** Write a program that checks if a given string is a palindrome (reads the same forwards and backwards) using a command-line

interface.

12. **Fibonacci Sequence Generator:** Create a program that generates the Fibonacci sequence up to a specified number of terms using a command-line interface.
13. **Prime Number Checker:** Develop a program that checks if a given number is a prime number or not using a command-line interface.
14. **Temperature Converter:** Build a program that converts temperatures between Celsius, Fahrenheit, and Kelvin using a command-line interface.
15. **BMI Calculator:** Write a program that calculates the Body Mass Index (BMI) using a command-line interface, given the height and weight of a person.
16. **Currency Converter:** Develop a program that converts between different currencies using a command-line interface, with support for multiple currencies and exchange rates.
17. **Binary to Decimal Converter:** Create a program that converts a binary number to its decimal equivalent using a command-line interface.
18. **Decimal to Binary Converter:** Write a program that converts a decimal number to its binary equivalent using a command-line interface.
19. **Simple File Encryption/Decryption:** Implement a program that can encrypt and decrypt files using a simple encryption algorithm, with options to specify the input and output files via the command line.
20. **Maze Solver:** Build a program that can solve a maze given as input using a command-line interface, with options to specify the maze layout and starting/ending points.
21. **Password Generator:** Write a program that generates random passwords of a specified length using a command-line interface, with options to specify the length and complexity of the passwords.
22. **Anagram Checker:** Create a program that checks if two given strings are anagrams of each other using a command-line interface.
23. **Binary Search Algorithm:** Implement the binary search algorithm to find a target value in a sorted array using a command-line interface.

24. **Sorting Algorithm Visualizer:** Develop a program that visualizes sorting algorithms (e.g., bubble sort, merge sort) using a command-line interface, with options to specify the array size and sorting algorithm.
25. **Chess Game:** Create a command-line chess game where two players can play against each other, with options to specify moves using algebraic notation.