KIET GROUP OF INSTITUTIONS
Connecting Life with Learning
www.kiet.edu
Delhi-NCR, Ghaziabad

Project Title: AI-Based Number Guessing Game

Student Name: Anwesha Singhal

Roll Number: 61

Course: CSEAI

Date: 11-03-2025

Introduction:

The AI-Based Number Guessing Game is an interactive program where the computer attempts to guess a number that the user is thinking of, using an optimized approach called the *Binary Search Algorithm*. Instead of randomly guessing numbers, the AI intelligently narrows down the range based on user feedback. This approach ensures that the game finds the correct number in the shortest possible time.

Methodology:

1. The user is asked to think of a number between 1 and 100.

2. The AI starts by guessing the middle number of the given range.

3. The user provides feedback:

   - If the guessed number is *too high*, the upper bound is reduced.

   - If the guessed number is *too low*, the lower bound is increased.

   - If the guessed number is *correct*, the game ends.

4. The AI repeats this process, halving the search space each time, ensuring that the number is found in at most *7 attempts* ($\log_2(100) \approx 7$). To visualize this, graphs are included in the program.

Code:

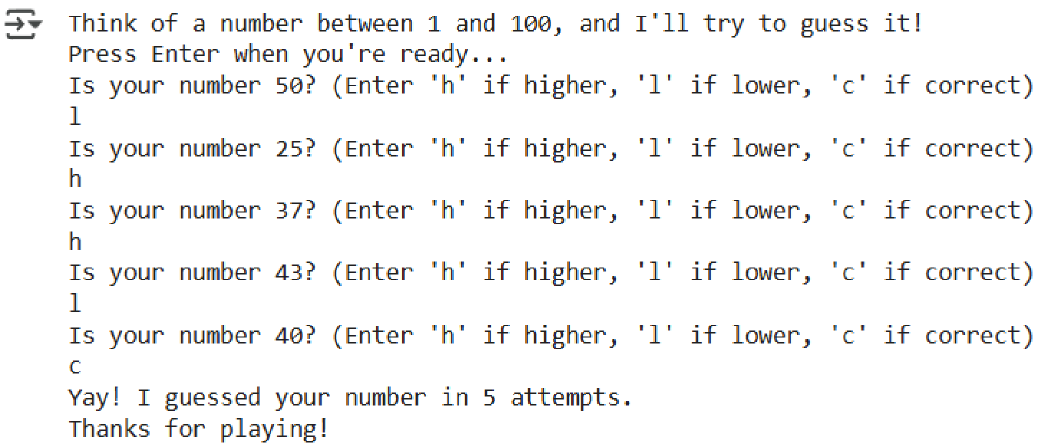
python

```
import random

import matplotlib.pyplot as plt
```

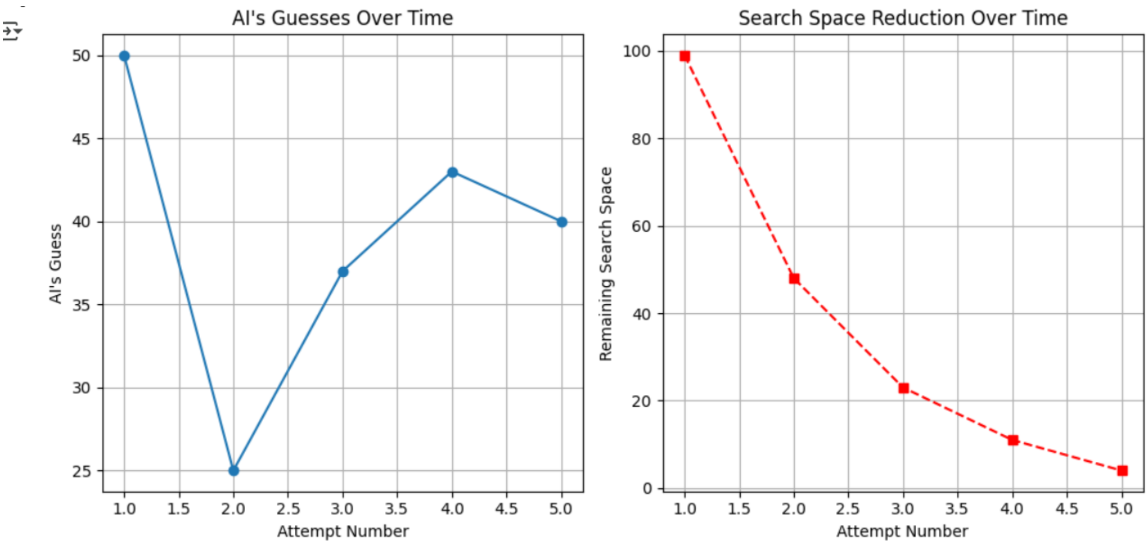
```
def ai_guess(low, high):
    """AI guesses a number intelligently using a binary
search approach"""
    return (low + high) // 2  # Middle of the current
range


def main():
    """Main function to play the AI-based Number
Guessing Game"""
    print("Think of a number between 1 and 100, and I'll
try to guess it!")
    input("Press Enter when you're ready...")  # Wait for
user to start the game


    low, high = 1, 100  # Initial range
    attempts = 0  # Count the number of attempts made
by AI
    guess_history = []  # Stores the guesses made by AI
    range_history = []  # Stores the remaining search
space size


    while low <= high:
        guess = ai_guess(low, high)  # AI makes a guess based on the current range
        guess_history.append(guess)
         range_history.append(high - low)


        print(f"Is your number {guess}? (Enter 'h' if higher, 'l' if lower, 'c' if correct)")
        response = input().strip().lower()  # Get user feedback and normalize input
        attempts += 1  # Increase attempt count


        if response == 'c':
            print(f"Yay! I guessed your number in {attempts} attempts.")
            break
        elif response == 'h':
            low = guess + 1  # Narrow the range upwards
        elif response == 'l':
            high = guess - 1  # Narrow the range downwards
        else:
            print("Invalid input. Please enter 'h', 'l', or 'c'.")
```

```
    print("Thanks for playing!")  # End of the game message

    # Plot the graphs
    plt.figure(figsize=(10, 5))

    # First graph: AI's guesses over attempts
    plt.subplot(1, 2, 1)
    plt.plot(range(1, len(guess_history) + 1), guess_history, marker='o', linestyle='-')
    plt.xlabel("Attempt Number")
    plt.ylabel("AI's Guess")
    plt.title("AI's Guesses Over Time")
    plt.grid(True)

    # Second graph: Remaining search space size over attempts
    plt.subplot(1, 2, 2)
    plt.plot(range(1, len(range_history) + 1), range_history, marker='s', linestyle='--', color='r')
    plt.xlabel("Attempt Number")
    plt.ylabel("Remaining Search Space")
    plt.title("Search Space Reduction Over Time")
    plt.grid(True)

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()  # Start the game when the script is run
```

Output/Result:



```
Think of a number between 1 and 100, and I'll try to guess it!
Press Enter when you're ready...
Is your number 50? (Enter 'h' if higher, 'l' if lower, 'c' if correct)
l
Is your number 25? (Enter 'h' if higher, 'l' if lower, 'c' if correct)
h
Is your number 37? (Enter 'h' if higher, 'l' if lower, 'c' if correct)
h
Is your number 43? (Enter 'h' if higher, 'l' if lower, 'c' if correct)
l
Is your number 40? (Enter 'h' if higher, 'l' if lower, 'c' if correct)
c
Yay! I guessed your number in 5 attempts.
Thanks for playing!
```



References/Credits:

- Concept of Binary Search Algorithm: []

- Code developed by: [Anwesha Singhal]

- Any additional images, datasets, or resources used in the report should be credited here.