```
import random

def generate_math_clue(secret_number, guess, clue_type):
    if clue_type == "parity":
        return f"The number is {'even' if secret_number % 2 == 0 else 'odd'}."
    elif clue_type == "divisibility":
        divisors = [i for i in range(1, 11) if secret_number % i == 0]
        return f"The number is divisible by {random.choice(divisors)}."
    elif clue_type == "digit_sum":
        digit_sum = sum(int(digit) for digit in str(secret_number))
        return f"The sum of the digits is {digit_sum}."
    elif clue_type == "prime":
        if secret_number < 2:
            return "The number is not prime."
        for i in range(2, int(secret_number**0.5) + 1):
            if secret_number % i == 0:
                return "The number is not prime."
        return "The number is prime."

def calculate_score(attempts, max_attempts, difficulty):
    base_score = 100
    if difficulty == "easy":
        return max(0, base_score - (attempts * 5))
    elif difficulty == "medium":
        return max(0, base_score - (attempts * 7))
    elif difficulty == "hard":
        return max(0, base_score - (attempts * 10))

def print_instructions():
    print("""
    Welcome to the Game Menu!
    Instructions for "Guess the Number":
    1. Choose a difficulty level: Easy, Medium, or Hard.
    2. The game will randomly select a number within the range based on difficulty.
    3. You will have a limited number of attempts to guess the number.
    4. After each guess, you will receive feedback whether your guess is too high or too low.
    5. Mathematical clues will be provided to help you guess the number.
    6. Try to guess the number in the fewest attempts to maximize your score.

    Instructions for "Hangman":
    1. Choose a category and difficulty level.
    2. You need to guess the letters in the word.
    3. You have a limited number of incorrect guesses.
    4. Try to guess the word before you run out of attempts.
    5. Scoring is based on the difficulty level and number of incorrect guesses.
    """)

def choose_difficulty(game):
```

```
    while True:
        print(f"Choose difficulty level for {game}:")
        print("1. Easy")
        print("2. Medium")
        print("3. Hard")
        choice = input("Enter 1, 2, or 3: ")

        if choice == '1':
            return 'easy'
        elif choice == '2':
            return 'medium'
        elif choice == '3':
            return 'hard'
        else:
            print("Invalid choice. Please try again.")

def guess_the_number():
    difficulty = choose_difficulty("Guess the Number")
    max_number = 200 if difficulty == 'hard' else (50 if difficulty == 'easy' else 100)
    max_attempts = 15 if difficulty == 'hard' else 10
    max_clues = 3 if difficulty == 'hard' else float('inf')

    secret_number = random.randint(1, max_number)
    attempts = 0
    clues_given = 0
    clue_types = ["parity", "divisibility", "digit_sum", "prime"]

    while attempts < max_attempts:
        try:
            guess = int(input(f"Enter your guess (1-{max_number}): "))
            if guess < 1 or guess > max_number:
                print(f"Please enter a number between 1 and {max_number}.")
                continue
        except ValueError:
            print("Invalid input. Please enter an integer.")
            continue

        attempts += 1

        if guess == secret_number:
            print(f"Congratulations! You guessed the number in {attempts} attempts. 👍")
            score = calculate_score(attempts, max_attempts, difficulty)
            print(f"Your score: {score}")
            break
        elif guess < secret_number:
            print("Too low!")
        else:
            print("Too high!")
```

```
        if clues_given < max_clues:
            clue_type = random.choice(clue_types)
            clue = generate_math_clue(secret_number, guess, clue_type)
            print(f"Clue: {clue}")
            clues_given += 1

        if attempts == max_attempts:
            print(f"Sorry, you've used all {max_attempts} attempts. The number was
{secret_number}. 👎")
            break

def get_hangman_display(incorrect_guesses):
    hangman_stages = [
        """
          ------
          |    |
          |
          |
          |
          |
        --------
        """,
        """
          ------
          |    |
          |    O
          |
          |
          |
        --------
        """,
        """
          ------
          |    |
          |    O
          |    |
          |
          |
        --------
        """,
        """
          ------
          |    |
          |    O
          |   /|
          |
          |
```

```
        --------
        """,
        """
          ------
          |    |
          |    O
          |   /|\\
          |
          |
        --------
        """,
        """
          ------
          |    |
          |    O
          |   /|\\
          |   /
          |
        --------
        """,
        """
          ------
          |    |
          |    O
          |   /|\\
          |   / \\
          |
        --------
        """
    ]
    return hangman_stages[incorrect_guesses]

def hangman():
    word_categories = {
        "Animals": ["monkey", "dog", "cat", "cheetah", "lion", "tiger", "elephant", "giraffe"],
        "Fruits": ["apple", "orange", "pineapple", "pomegranate", "strawberry", "watermelon",
"banana", "mango"],
        "Countries": ["india", "pakistan", "china", "america", "japan", "australia", "germany",
"switzerland"]
    }

    categories = list(word_categories.keys())

    print("Choose a category for Hangman:")
    for i, category in enumerate(categories, 1):
        print(f"{i}. {category}")
    category_choice = int(input("Enter the number of the chosen category: ")) - 1
```

```
if category_choice < 0 or category_choice >= len(categories):
    print("Invalid choice. Please choose a valid category.")
    return

chosen_category = categories[category_choice]
difficulty = choose_difficulty("Hangman")
word_list = word_categories[chosen_category]
secret_word = random.choice(word_list)
guessed_letters = set()
incorrect_guesses = 0
max_incorrect_guesses = 6
display_word = ['_'] * len(secret_word)

print(f"Welcome to Hangman! The category is: {chosen_category}")

while incorrect_guesses < max_incorrect_guesses:
    print(get_hangman_display(incorrect_guesses))
    print(" ".join(display_word))

    guess = input("Guess a letter: ").lower()

    if len(guess) != 1 or not guess.isalpha():
        print("Invalid input. Please enter a single letter.")
        continue

    if guess in guessed_letters:
        print("You've already guessed that letter. Try again.")
        continue

    guessed_letters.add(guess)

    if guess in secret_word:
        for i, letter in enumerate(secret_word):
            if letter == guess:
                display_word[i] = guess
        print("Correct guess!")
    else:
        incorrect_guesses += 1
        print("Incorrect guess.")

    if "_" not in display_word:
        print(get_hangman_display(incorrect_guesses))
        print(f"Congratulations! You guessed the word '{secret_word}' correctly. 👍")
        score = calculate_score(incorrect_guesses, max_incorrect_guesses, difficulty)
        print(f"Your score: {score}")
        break

if incorrect_guesses == max_incorrect_guesses:
```

```
        print(get_hangman_display(incorrect_guesses))
        print(f"Sorry, you've used all your attempts. The word was '{secret_word}'. 👎")

def main_menu():
    while True:
        print("Welcome to the Game Menu!")
        print("1. Guess the Number")
        print("2. Hangman")
        print("3. Instructions")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            guess_the_number()
        elif choice == '2':
            hangman()
        elif choice == '3':
            print_instructions()
        elif choice == '4':
            print("Thank you for playing! Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
                        ■       main_menu()
                        ■
```