Project Title:

Prime Numbers Finder and Saver

Objective:

To create a C program that calculates all prime numbers in a user-defined range and saves them to a text file for future reference.

Features:

1. Accept input for the range (start and end values).
2. Calculate all prime numbers in the range using efficient logic.
3. Save the prime numbers to a file (`primes.txt`).
4. Display the prime numbers to the user.

Logic of the Program:

1. **Definition of Prime Number**: A number is prime if it is greater than 1 and divisible only by 1 and itself.
2. **Optimization**:
    - Numbers divisible by any value less than or equal to their square root are non-prime.
    - Skip even numbers (except 2).
3. **File Handling**:
    - Open a text file in write mode to store the calculated prime numbers.
4. **Flow**:
    - Accept the range as input.
    - Use a function to check whether each number in the range is prime.
    - Save the prime numbers to a file and display them on the screen.

Code:

```
#include <stdio.h>
#include <math.h>

// Function prototypes
int isPrime(int num);
void findPrimesInRange(int start, int end, const char *filename);

int main() {
```

```
    int start, end;
    char filename[] = "primes.txt";

    printf("Prime Numbers Finder\n");
    printf("Enter the start of the range: ");
    scanf("%d", &start);
    printf("Enter the end of the range: ");
    scanf("%d", &end);

    if (start > end || start < 2) {
        printf("Invalid range! Ensure start >= 2 and start <= end.\n");
        return 1;
    }

    findPrimesInRange(start, end, filename);

    printf("Prime numbers have been saved to '%s'.\n", filename);
    return 0;
}

// Function to check if a number is prime
int isPrime(int num) {
    if (num < 2) return 0; // Numbers less than 2 are not prime
    if (num == 2) return 1; // 2 is prime
    if (num % 2 == 0) return 0; // Even numbers greater than 2 are not prime

    for (int i = 3; i <= sqrt(num); i += 2) {
        if (num % i == 0) {
            return 0; // Number is not prime
        }
    }
    return 1; // Number is prime
}

// Function to find and save prime numbers in a range
void findPrimesInRange(int start, int end, const char *filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error: Could not open file '%s' for writing.\n", filename);
        return;
    }

    printf("Prime numbers in the range [%d, %d]:\n", start, end);
    for (int num = start; num <= end; num++) {
```

```
        if (isPrime(num)) {
            printf("%d ", num);
            fprintf(file, "%d\n", num); // Save to file
        }
    }
    printf("\n");
    fclose(file);
}
```

Explanation of the Code:

1. **Prime Check (`isPrime` Function)**:
    - Numbers less than 2 are non-prime.
    - For numbers greater than 2, check divisibility only up to their square root.
    - Use `i += 2` to skip even numbers after 2 for efficiency.
2. **Finding Primes in Range (`findPrimesInRange` Function)**:
    - Iterates through all numbers in the specified range.
    - Calls `isPrime` for each number to determine if it's prime.
    - Prints prime numbers to the console and writes them to a file.
3. **File Handling**:
    - Opens `primes.txt` in write mode to save prime numbers.
    - Closes the file after writing.

How to Run the Program:

1. **Compile**: Use a C compiler to compile the code, e.g., `gcc prime_finder.c -o prime_finder`.
2. **Execute**: Run the program: `./prime_finder`.
3. **Provide Input**: Enter the start and end of the range.
4. **Output**: The prime numbers will be displayed on the console and saved in `primes.txt`.

Output Sample:

Input:

Enter the start of the range: 10
Enter the end of the range: 50

Console Output:

Prime Numbers Finder
Prime numbers in the range [10, 50]:
11 13 17 19 23 29 31 37 41 43 47
Prime numbers have been saved to 'primes.txt'.

File (`primes.txt`):

11
13
17
19
23
29
31
37
41
43
47

Key Concepts Covered:

1. Efficient prime-checking logic with square root optimization.
2. File handling for saving results.
3. User-friendly interface for input and output.
4. Modular design with functions for prime-checking and saving.
5. This project is an excellent demonstration of control structures, mathematical logic, and file handling in C programming. It can be further enhanced by supporting additional features like user-defined filenames or progress tracking for large ranges.