Mod 5 assignment

Question;

Create a Bash script 'file_analyzer.sh', to demonstrate the following concepts:

1. Recursive functions

- Write a recursive function to search for files in a directory and its subdirectories containing a specific keyword.

2. Redirection and error handling

- Log errors (e.g., invalid arguments, missing files) to 'errors.log' and display them in the terminal.

3. Here document and here string

- Use a here document to display a help menu when the '--help' option is passed.
- Search for a keyword in a specified file using a here string.

4. Special parameters

- Use parameters like '$0', '$#', '$?' and '$@' to provide meaningful feedback.

5. Regular expressions

- Validate inputs with regular expressions (Check if the file exists and the keyword is not empty and valid).

6. Command-line arguments using getopts

- Use 'getopts' to handle:
  '-d <directory>': Directory to search.
  '-k <keyword>': Keyword to search.
  '-f <file>': File to search directly.
  '--help': Display the help menu.

Answer:

```
#!/bin/bash

ERROR_LOG="errors.log"

> "$ERROR_LOG"

log_error() {
    local msg="[ERROR] $(date '+%Y-%m-%d %H:%M:%S') - $1"
    echo "$msg" | tee -a "$ERROR_LOG" >&2
}

show_help() {
    cat <<HELP_MENU
============================================================
  FILE ANALYZER - Help Menu
============================================================

  Script Name : $0
  Description : Search for keywords in files or directories.

  USAGE:
    $0 [OPTIONS]

  OPTIONS:
    -d <directory>   Directory to search recursively
    -k <keyword>     Keyword to search for (required)
    -f <file>        Search for keyword in a specific file
    --help           Display this help menu

  EXAMPLES:
    $0 -d logs -k error
    $0 -f script.sh -k TODO
    $0 --help

  NOTES:
    - Either -d or -f must be provided along with -k.
    - The keyword must be a non-empty, valid string.
============================================================
HELP_MENU
}

validate_keyword() {
    local keyword="$1"
```

```
    if [[ -z "$keyword" ]]; then
        log_error "Keyword is empty. Please provide a valid keyword."
        return 1
    fi

    if [[ ! "$keyword" =~ ^[a-zA-Z0-9_.\ -]+$ ]]; then
        log_error "Keyword '$keyword' contains invalid characters. Allowed: letters,
digits, _ - . and spaces."
        return 1
    fi

    return 0
}

validate_file() {
    local file="$1"

    if [[ ! -e "$file" ]]; then
        log_error "File '$file' does not exist."
        return 1
    fi

    if [[ ! -f "$file" ]]; then
        log_error "'$file' is not a regular file."
        return 1
    fi

    if [[ ! -r "$file" ]]; then
        log_error "File '$file' is not readable."
        return 1
    fi

    return 0
}

search_in_file() {
    local file="$1"
    local keyword="$2"

    validate_file "$file"
    if [[ $? -ne 0 ]]; then
        return 1
    fi

    echo "--- Searching in file: $file ---"

    local content
    content=$(<"$file")
```

```
    local results
    results=$(grep -n "$keyword" <<< "$content")

    if [[ -n "$results" ]]; then
        echo "$results"
        return 0
    else
        echo "  (no matches found)"
        return 1
    fi
}

search_directory() {
    local dir="$1"
    local keyword="$2"
    local match_count=0

    if [[ ! -d "$dir" ]]; then
        log_error "Directory '$dir' does not exist or is not a directory."
        return 1
    fi

    if [[ ! -r "$dir" ]]; then
        log_error "Directory '$dir' is not readable."
        return 1
    fi

    for item in "$dir"/*; do

        [[ -e "$item" ]] || continue

        if [[ -d "$item" ]]; then
            search_directory "$item" "$keyword"
            match_count=$((match_count + $?))

        elif [[ -f "$item" && -r "$item" ]]; then
            local content
            content=$(<"$item") 2>/dev/null
            local results
            results=$(grep -n "$keyword" <<< "$content" 2>/dev/null)

            if [[ -n "$results" ]]; then
                echo "--- $item ---"
                echo "$results"
                echo ""
                match_count=$((match_count + 1))
            fi
        fi
```

```
    done

    return $match_count
}

echo "Script: $0"
echo "Total arguments received: $#"
echo "Arguments: $@"
echo ""

if [[ $# -eq 0 ]]; then
    log_error "No arguments provided. Use --help for usage information."
    echo ""
    show_help
    exit 1
fi

for arg in "$@"; do
    if [[ "$arg" == "--help" ]]; then
        show_help
        exit 0
    fi
done

DIRECTORY=""
KEYWORD=""
FILE=""

while getopts ":d:k:f:" opt; do
    case "$opt" in
        d)
            DIRECTORY="$OPTARG"
            ;;
        k)
            KEYWORD="$OPTARG"
            ;;
        f)
            FILE="$OPTARG"
            ;;
        :)
            log_error "Option -$OPTARG requires an argument."
            exit 1
            ;;
        \?)
            log_error "Invalid option: -$OPTARG. Use --help for usage."
            exit 1
            ;;
    esac
```

```
done

if [[ -z "$KEYWORD" ]]; then
    log_error "Keyword (-k) is required. Use --help for usage."
    exit 1
fi

validate_keyword "$KEYWORD"
if [[ $? -ne 0 ]]; then
    exit 1
fi

if [[ -z "$DIRECTORY" && -z "$FILE" ]]; then
    log_error "You must specify either a directory (-d) or a file (-f). Use --help for
usage."
    exit 1
fi

if [[ -n "$FILE" ]]; then
    echo "========================================"
    echo " Searching file '$FILE' for keyword: '$KEYWORD'"
    echo "========================================"
    echo ""
    search_in_file "$FILE" "$KEYWORD"
    exit_code=$?
    if [[ $exit_code -eq 0 ]]; then
        echo ""
        echo "Search completed successfully (exit code: $exit_code)."
    else
        echo ""
        echo "Search completed with no matches (exit code: $exit_code)."
    fi
fi

if [[ -n "$DIRECTORY" ]]; then
    echo "========================================"
    echo " Recursively searching '$DIRECTORY' for keyword: '$KEYWORD'"
    echo "========================================"
    echo ""
    search_directory "$DIRECTORY" "$KEYWORD"
    exit_code=$?
    if [[ $exit_code -gt 0 ]]; then
        echo "Search completed. Files with matches: $exit_code"
    else
        echo "Search completed. No matches found."
    fi
fi
```

```
echo ""
echo "Errors (if any) have been logged to '$ERROR_LOG'."
exit 0
```

```
vishnu@vishnu:~/Documents/linux_assignments/assignment5$ ./file_analyzer.sh -f new.txt -k bad
Script: ./file_analyzer.sh
Total arguments received: 4
Arguments: -f new.txt -k bad

========================================
 Searching file 'new.txt' for keyword: 'bad'
========================================

--- Searching in file: new.txt ---
1:error can be bad

Search completed successfully (exit code: 0).

Errors (if any) have been logged to 'errors.log'.
```

```
vishnu@vishnu:~/Documents/linux_assignments/assignment5$ ./file_analyzer.sh --help
Script: ./file_analyzer.sh
Total arguments received: 1
Arguments: --help

============================================================
  FILE ANALYZER - Help Menu
============================================================

  Script Name : ./file_analyzer.sh
  Description : Search for keywords in files or directories.

  USAGE:
    ./file_analyzer.sh [OPTIONS]

  OPTIONS:
    -d <directory>   Directory to search recursively
    -k <keyword>     Keyword to search for (required)
    -f <file>        Search for keyword in a specific file
    --help           Display this help menu

  EXAMPLES:
    ./file_analyzer.sh -d logs -k error
    ./file_analyzer.sh -f script.sh -k TODO
    ./file_analyzer.sh --help

  NOTES:
    - Either -d or -f must be provided along with -k.
    - The keyword must be a non-empty, valid string.
============================================================
```