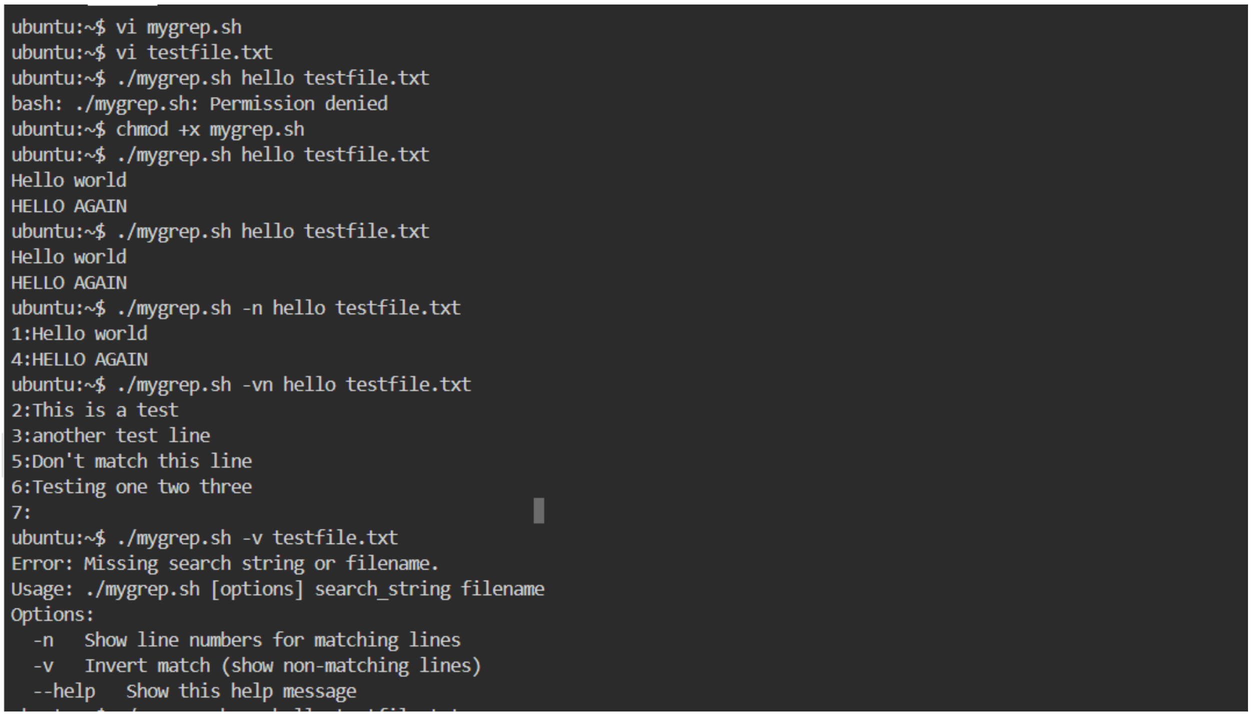
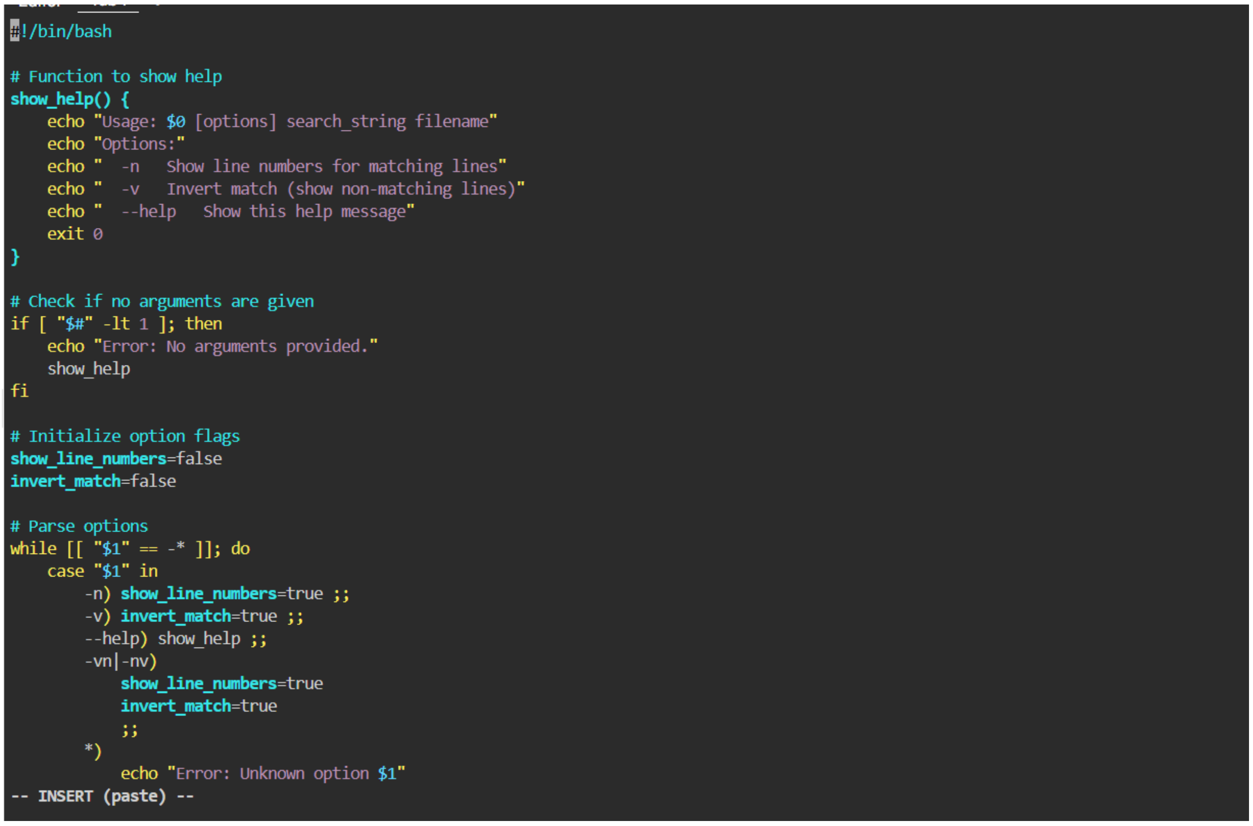
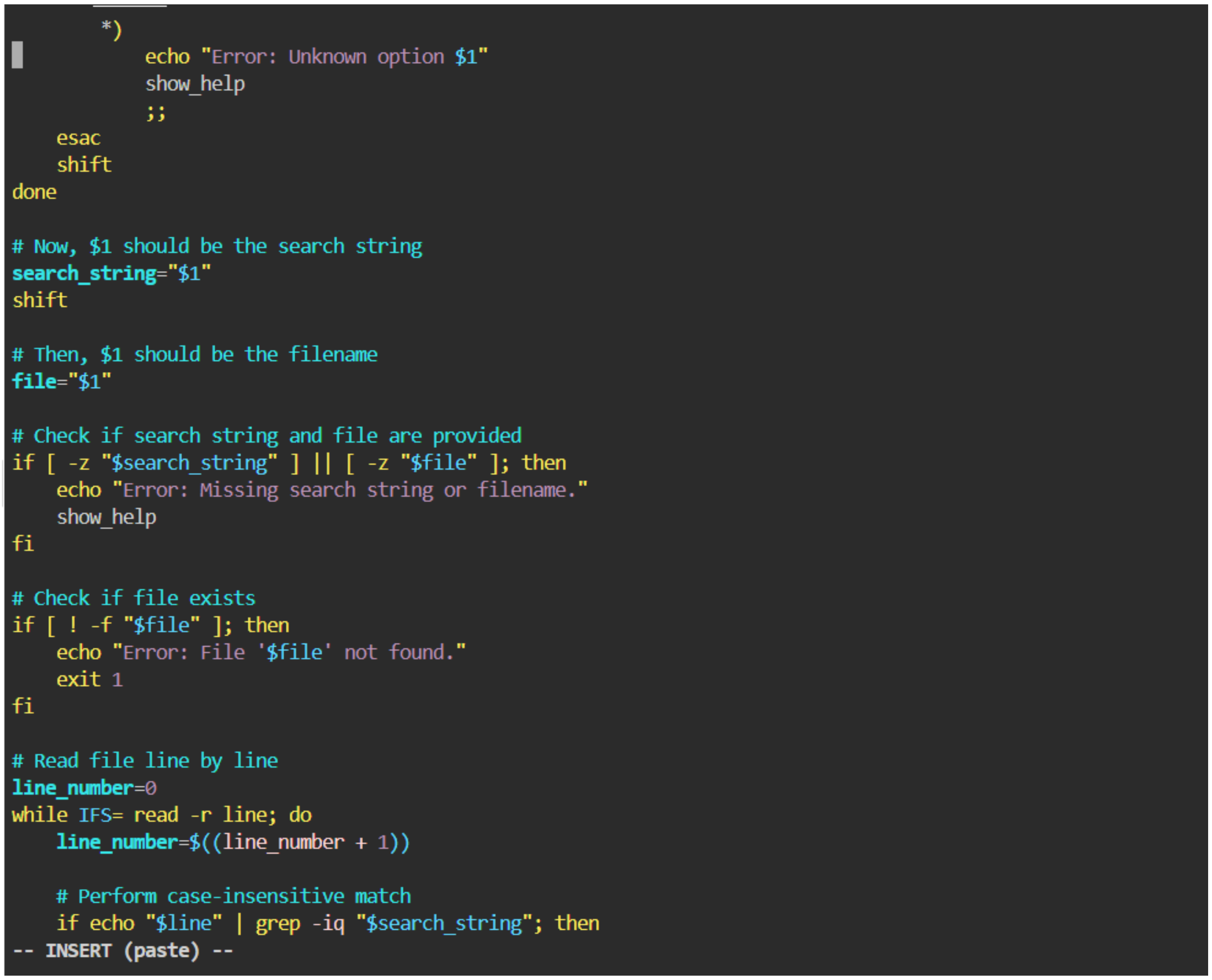


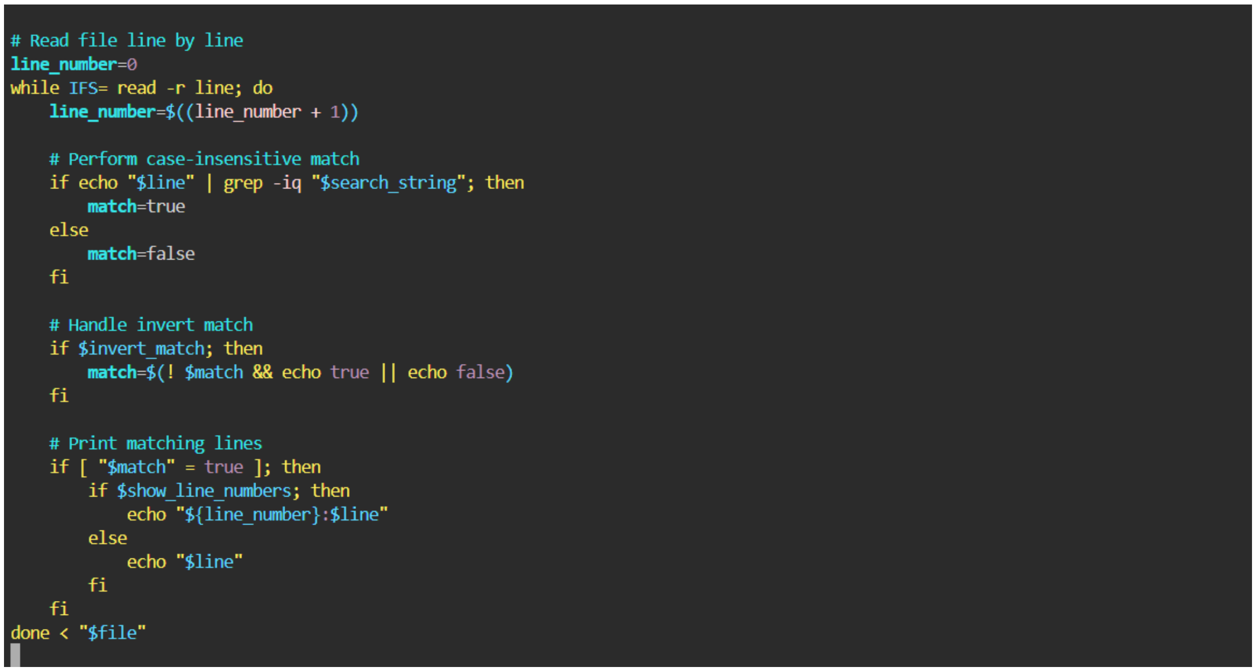
```
ubuntu:~$ vi mygrep.sh
ubuntu:~$ vi testfile.txt
ubuntu:~$ ./mygrep.sh hello testfile.txt
bash: ./mygrep.sh: Permission denied
ubuntu:~$ chmod +x mygrep.sh
ubuntu:~$ ./mygrep.sh hello testfile.txt
Hello world
HELLO AGAIN
ubuntu:~$ ./mygrep.sh hello testfile.txt
Hello world
HELLO AGAIN
ubuntu:~$ ./mygrep.sh -n hello testfile.txt
1:Hello world
4:HELLO AGAIN
ubuntu:~$ ./mygrep.sh -vn hello testfile.txt
2:This is a test
3:another test line
5:Don't match this line
6:Testing one two three
7:
ubuntu:~$ ./mygrep.sh -v testfile.txt
Error: Missing search string or filename.
Usage: ./mygrep.sh [options] search_string filename
Options:
  -n    Show line numbers for matching lines
  -v    Invert match (show non-matching lines)
  --help    Show this help message
```



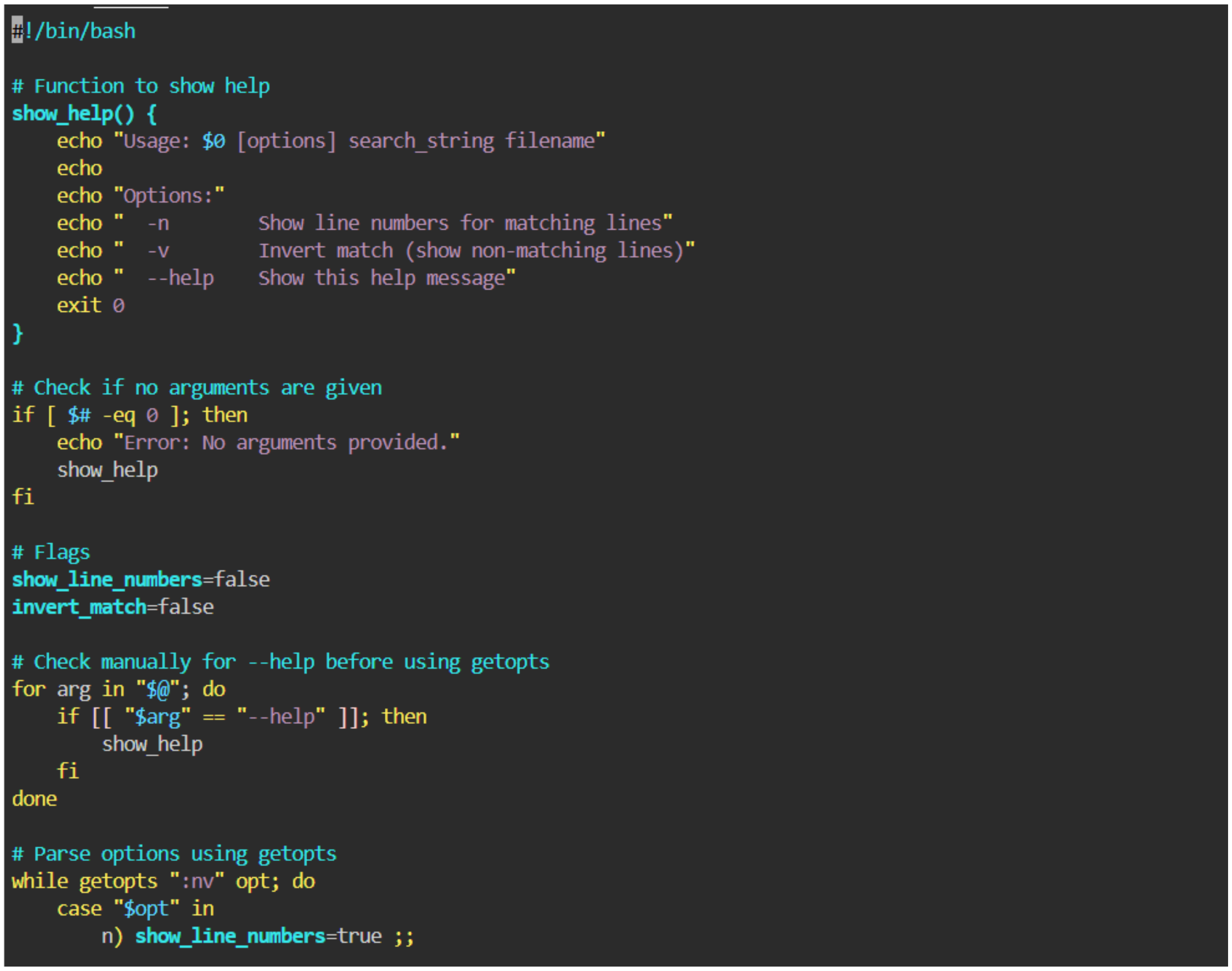
```
#!/bin/bash

# Function to show help
show_help() {
    echo "Usage: $0 [options] search_string filename"
    echo "Options:"
    echo "  -n    Show line numbers for matching lines"
    echo "  -v    Invert match (show non-matching lines)"
    echo "  --help    Show this help message"
    exit 0
}

# Check if no arguments are given
if [ "$#" -lt 1 ]; then
    echo "Error: No arguments provided."
    show_help
fi

# Initialize option flags
show_line_numbers=false
invert_match=false

# Parse options
while [[ "$1" == -* ]]; do
    case "$1" in
        -n) show_line_numbers=true ;;
        -v) invert_match=true ;;
        --help) show_help ;;
        -vn|-nv)
            show_line_numbers=true
            invert_match=true
            ;;
        *)
            echo "Error: Unknown option $1"
-- INSERT (paste) --
```



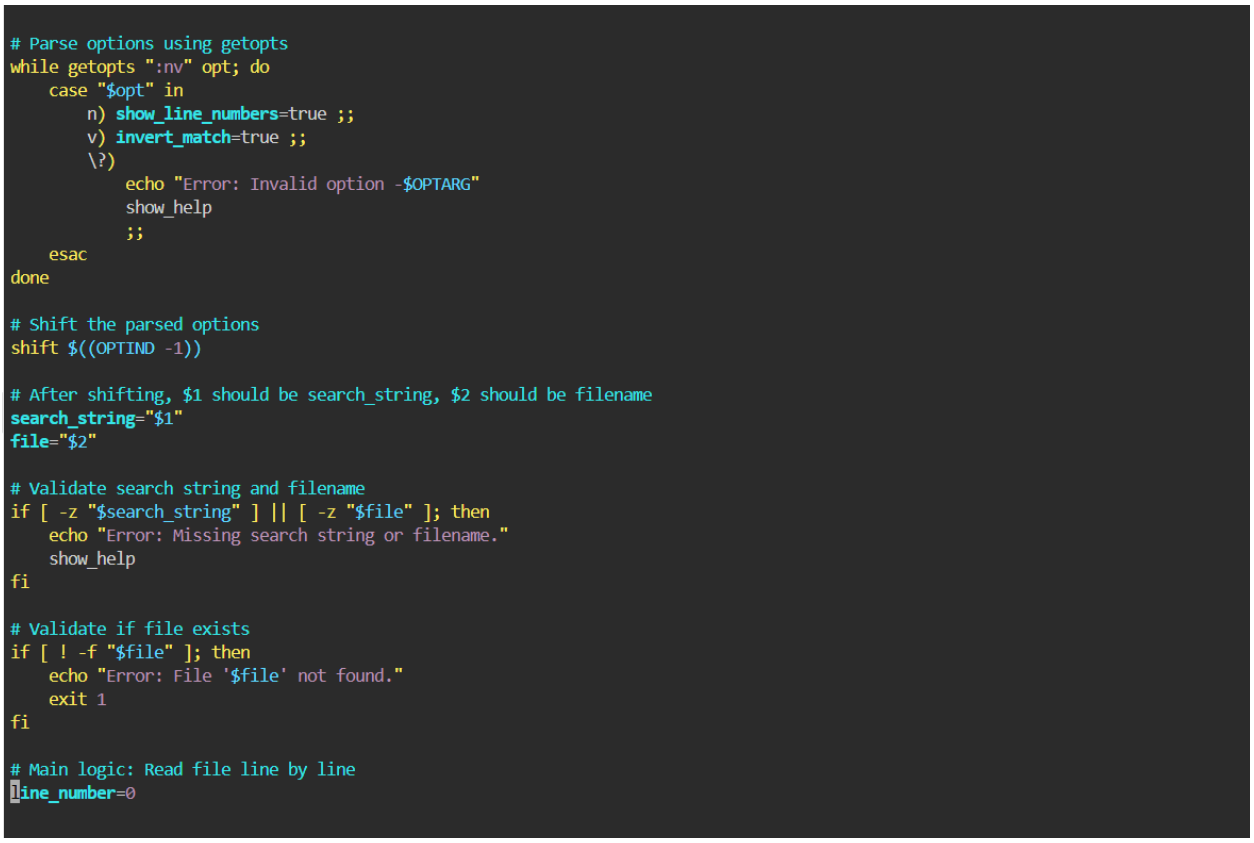
```
        *)
            echo "Error: Unknown option $1"
            show_help
            ;;
    esac
    shift
done

# Now, $1 should be the search string
search_string="$1"
shift

# Then, $1 should be the filename
file="$1"

# Check if search string and file are provided
if [ -z "$search_string" ] || [ -z "$file" ]; then
    echo "Error: Missing search string or filename."
    show_help
fi

# Check if file exists
if [ ! -f "$file" ]; then
    echo "Error: File '$file' not found."
    exit 1
fi

# Read file line by line
line_number=0
while IFS= read -r line; do
    line_number=$((line_number + 1))

    # Perform case-insensitive match
    if echo "$line" | grep -iq "$search_string"; then
-- INSERT (paste) --
```



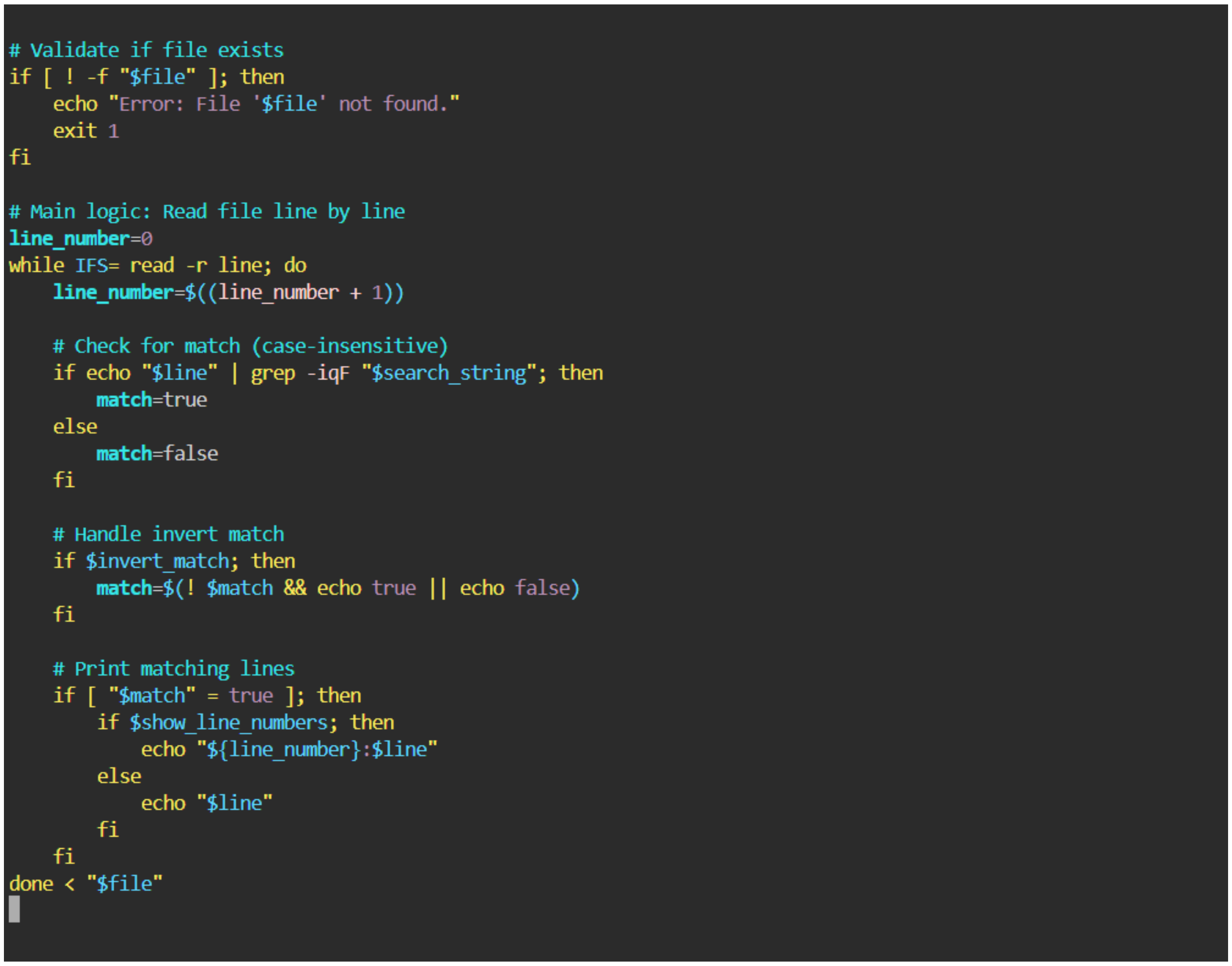
```bash
# Read file line by line
line_number=0
while IFS= read -r line; do
    line_number=$((line_number + 1))

    # Perform case-insensitive match
    if echo "$line" | grep -iq "$search_string"; then
        match=true
    else
        match=false
    fi

    # Handle invert match
    if $invert_match; then
        match=$(! $match && echo true || echo false)
    fi

    # Print matching lines
    if [ "$match" = true ]; then
        if $show_line_numbers; then
            echo "${line_number}:$line"
        else
            echo "$line"
        fi
    fi
done < "$file"
```



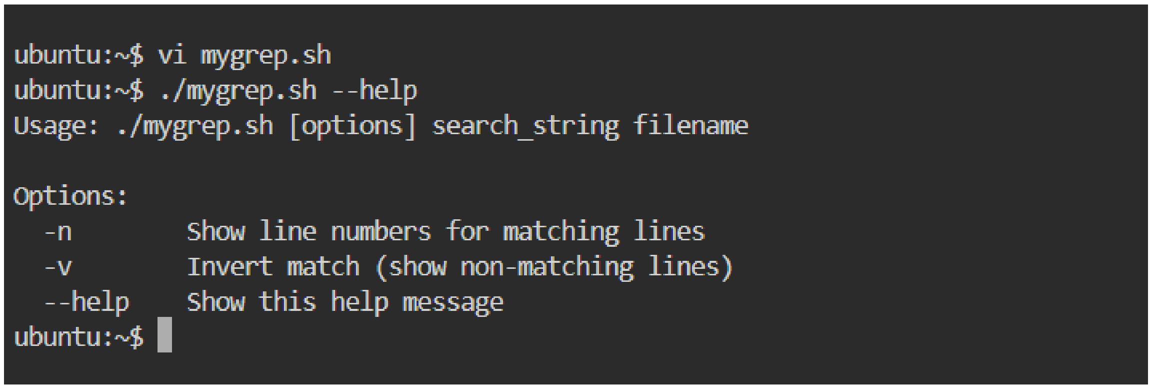
```bash
#!/bin/bash

# Function to show help
show_help() {
    echo "Usage: $0 [options] search_string filename"
    echo
    echo "Options:"
    echo "  -n        Show line numbers for matching lines"
    echo "  -v        Invert match (show non-matching lines)"
    echo "  --help    Show this help message"
    exit 0
}

# Check if no arguments are given
if [ $# -eq 0 ]; then
    echo "Error: No arguments provided."
    show_help
fi

# Flags
show_line_numbers=false
invert_match=false

# Check manually for --help before using getopts
for arg in "$@"; do
    if [[ "$arg" == "--help" ]]; then
        show_help
    fi
done

# Parse options using getopts
while getopts ":nv" opt; do
    case "$opt" in
        n) show_line_numbers=true ;;
```



```
# Parse options using getopts
while getopts ":nv" opt; do
    case "$opt" in
        n) show_line_numbers=true ;;
        v) invert_match=true ;;
        \?)
            echo "Error: Invalid option -$OPTARG"
            show_help
            ;;
    esac
done

# Shift the parsed options
shift $((OPTIND -1))

# After shifting, $1 should be search_string, $2 should be filename
search_string="$1"
file="$2"

# Validate search string and filename
if [ -z "$search_string" ] || [ -z "$file" ]; then
    echo "Error: Missing search string or filename."
    show_help
fi

# Validate if file exists
if [ ! -f "$file" ]; then
    echo "Error: File '$file' not found."
    exit 1
fi

# Main logic: Read file line by line
line_number=0
```



```
# Validate if file exists
if [ ! -f "$file" ]; then
    echo "Error: File '$file' not found."
    exit 1
fi

# Main logic: Read file line by line
line_number=0
while IFS= read -r line; do
    line_number=$((line_number + 1))

    # Check for match (case-insensitive)
    if echo "$line" | grep -iqF "$search_string"; then
        match=true
    else
        match=false
    fi

    # Handle invert match
    if $invert_match; then
        match=$(! $match && echo true || echo false)
    fi

    # Print matching lines
    if [ "$match" = true ]; then
        if $show_line_numbers; then
            echo "${line_number}:$line"
        else
            echo "$line"
        fi
    fi
done < "$file"
```



```
ubuntu:~$ vi mygrep.sh
ubuntu:~$ ./mygrep.sh --help
Usage: ./mygrep.sh [options] search_string filename

Options:
  -n        Show line numbers for matching lines
  -v        Invert match (show non-matching lines)
  --help    Show this help message
ubuntu:~$
```

1. How the script handles arguments and options:

The script first checks if --help is requested, then it uses getopts to parse -n and -v options.

After that, it expects two main inputs: the search string and the filename.

It validates inputs and adjusts behavior based on selected options:

-n shows line numbers.

-v shows non-matching lines.

Both options can work together.

If inputs are wrong or missing, it shows an error and usage help.

---

2. How I would add regex, -i, -c, -l options:

I would extend getopts to handle the new options and adjust the matching part:

-i would ignore case completely.

-c would count matches instead of showing lines.

-l would just show filenames with matches.

I might split features into separate functions for clarity.

---

3. Hardest part:

The most challenging part was handling multiple option combinations cleanly and making sure error messages guide the user properly.

Using getopts helped organize options better but needed careful handling.

---