Custom Command (mygrep.sh)

1- A breakdown of how your script handles arguments and options

Basically, we can divide the script and logic into four parts:

Initialization
I defined the placeholders and flags to be used later throughout the script.

```
#!/bin/bash

# options' flags and arguments placeholders
num=false
inv=false
sh_string=""
file=""
```

Parsing the Arguments
This is considered the heart of the script, where I defined the parsing logic. I followed a straightforward and not complicated method using `while` and `if` conditions, although it might not be the most effective way (as using `getopts`, for example, would be).
The idea here is to start with `$1` (the first argument after `$0`, the script name — as we know, the shell treats space as a separator between parts of commands), then check every possible option and turn flags on accordingly. At the end, we check for non-options using `-z` and assign placeholders for `file` and `search_string`.

```
# parse arguments
index=1
while [ $index -le $# ]; do
  current="${!index}"

  if [ "$current" = "--help" ]; then
    echo "Usage: $0 [-n] [-v] search_string file"
    exit 0

  elif [ "$current" = "-vn" ] || [ "$current" = "-nv" ]; then
    inv=true
    num=true

  elif [ "$current" = "-n" ]; then
    num=true
  elif [ "$current" = "-v" ]; then
    inv=true

  else
    # first non-option is search_string, second is file
    if [ -z "$sh_string" ]; then
      sh_string="$current"
    elif [ -z "$file" ]; then
      file="$current"
    fi
  fi

  index=$((index + 1))
```

Validation
This part concerns ensuring that the arguments exist. I used simple `if` logic here, although I

am aware it is not very powerful and can be easily fooled by certain test cases, as it doesn't determine exactly what is missing — is it the search string or the file?
The idea was to keep things simple and quick in this version. That is not the case, by the way, in my `mygrep_v2.sh`.

```
38
39    # make sure both search_string and file exists
40    if [ -z "$sh_string" ] || [ -z "$file" ]; then
41      echo "Error: Missing search_string or file"
42      echo "Usage: $0 [-n] [-v] search_string file"
43      exit 1
44    fi
45
46    # verify the file exists
47    if [ ! -f "$file" ]; then
48      echo "File not found: $file"
49      exit 1
50    fi
```

Building the Grep Command

This last part concerns gluing it all together using the original `grep` command.

```
51
52    cmd="grep -i"
53    [ "$inv" = true ] && cmd="$cmd -v"
54    [ "$num" = true ] && cmd="$cmd -n"
55
56    # run the search
57    $cmd "$sh_string" "$file"
```

2- A short paragraph: If you were to support regex or -i/-c/-l options, how would your structure change?

Well, the first thing that comes to mind is that it wouldn't be effective to use `if` conditions and test for every option individually. Maybe something like `case` would work better here, combined directly with the original `grep`.
The script would actually be much shorter.
Second, I wouldn't use placeholders for each option like `pat` or `num`; instead, I would collect options dynamically into a single placeholder, like `options`, which I would use in the `while` loop.

3- What part of the script was hardest to implement and why?

There was nothing especially hard.
Maybe the validation part was a little tricky (because of the logic I used), as I just used `-z` checks, which makes it hard to explain to the user what exactly is missing.

Abdalla Hammouda