

# Safe Edit Bash and Batch Scripts

**SID - 2408078**

<b>Planning.....</b>	<b>2</b>
What is the program?.....	2
Pseudocode.....	2
Flow Charts.....	4
<b>Linux Bash Script.....</b>	<b>7</b>
Code.....	7
Tests.....	11
<b>Windows Batch Script.....</b>	<b>18</b>
Code.....	18
Tests.....	22

# Planning

## What is the program?

The program allows the user to safely edit a file in the command line. The user opens a file either by calling the program with a file path as input or calling the program without then entering the file path. The program then creates a .bak backup copy and stores it in the backup directory. The program also writes a log of the backup.

## Pseudocode

BEGIN

    FUNCTION backupFile(filePath):

        # Create timestamp for the backup operation

        SET timestamp to the current date and time in the format "YYYY-MM-DD HH:MM:SS"

        SET backupFolderDir to chosen backup folder directory

        If backupFolder does not exist:

            CREATE folder at backupFolderDir

        ENDIF

        # Check if the file exists

        IF filePath does not exist:

            PRINT "Error: filePath does not exist"

            RETURN 1 # Exit the function if file does not exist

        END IF

        SET backupFileName to the filePath with its extension replaced by .bak

        COPY filePath to backupFileName

        SET numberOfLines to the number of lines in backupLog.txt

        IF numberOfLines equals 5:

            REMOVE the first line from backupLog.txt

        END IF

```
    APPEND timestamp + "Backup created:" + filePath + " → " + backupFileName + "to  
backupLog.txt"
```

```
    PRINT "Backup of filePath created successfully at timestamp"
```

```
END FUNCTION
```

```
FUNCTION edit(filePath):
```

```
    IF filePath exists:
```

```
        CALL backupFile(filePath)
```

```
        OPEN filePath in vim editor
```

```
    ELSE
```

```
        PRINT "Error: filePath does not exist in the current directory."
```

```
    END IF
```

```
END FUNCTION
```

```
    PRINT "Welcome to the Safe Edit Bash Script!"
```

```
    PRINT "This script will allow you to safely edit a file with backup and logging."
```

```
    IF exactly 1 command-line argument is given:
```

```
        CALL edit() with the argument passed
```

```
    ELSE IF more than 1 argument is given:
```

```
        PRINT "Error: Too many parameters entered."
```

```
    ELSE:
```

```
        PRINT "Which file would you like to edit?"
```

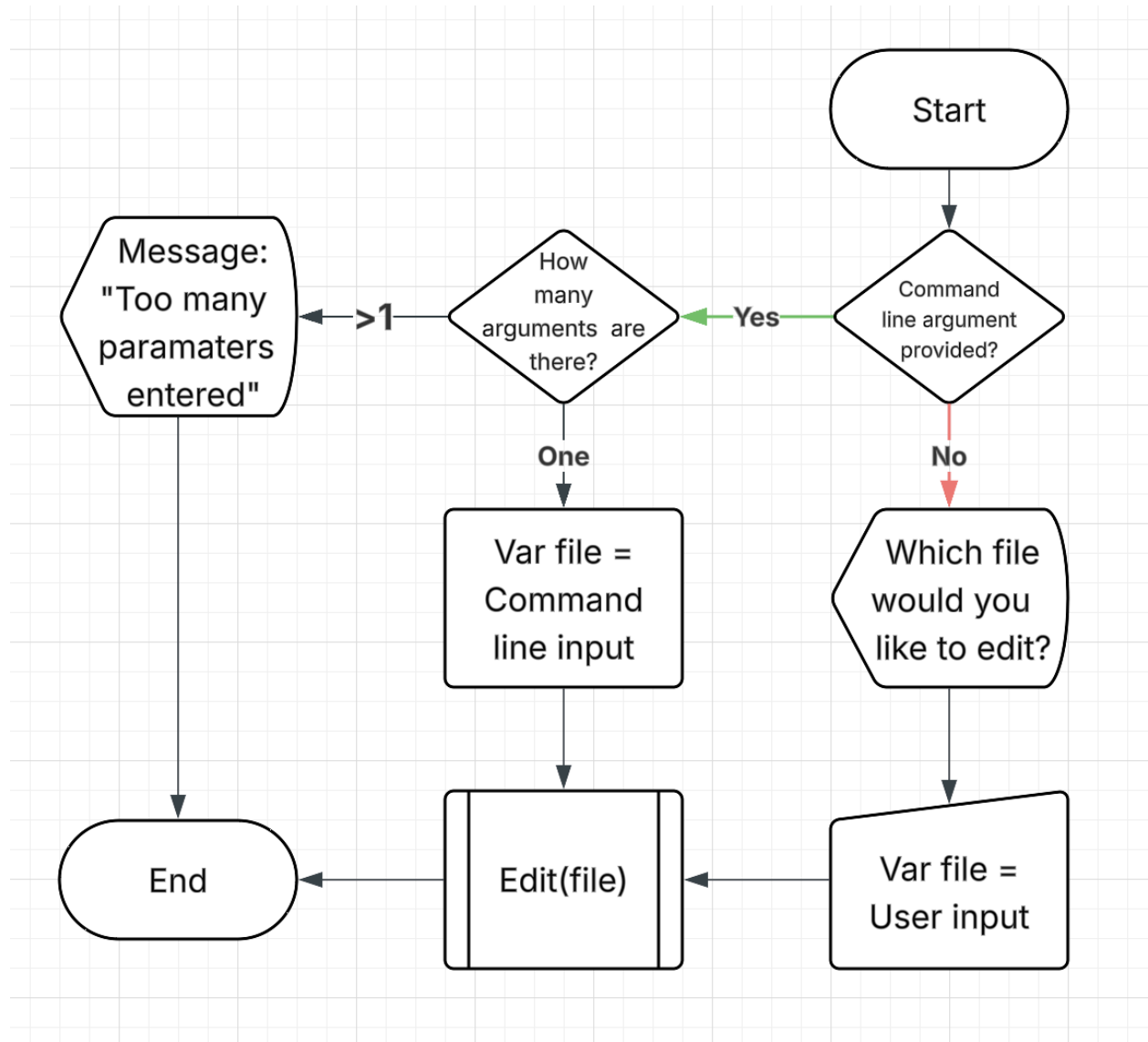
```
        READ filePath from user input
```

```
        CALL edit with the user-provided filePath
```

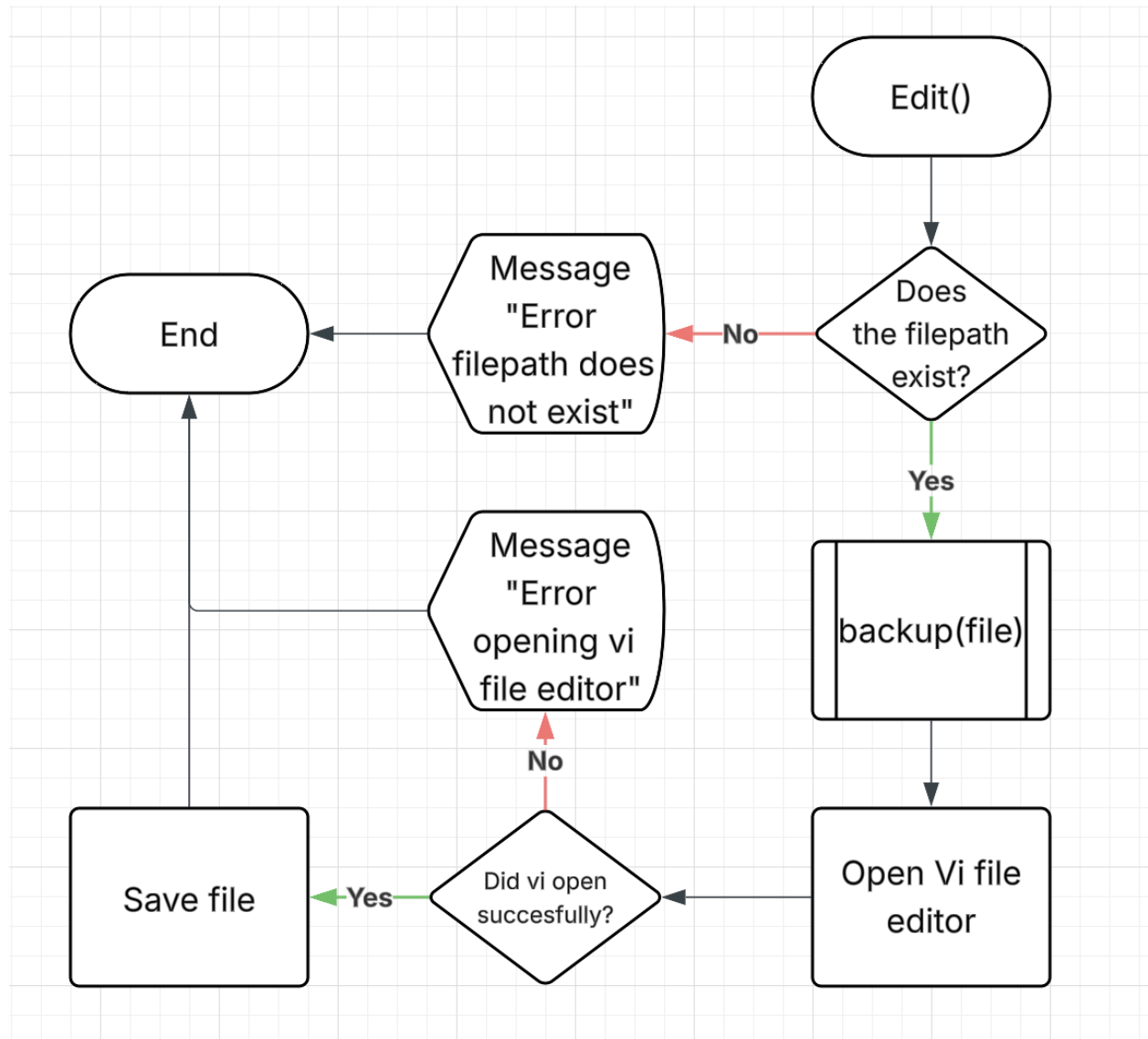
```
    END IF
```

```
END
```

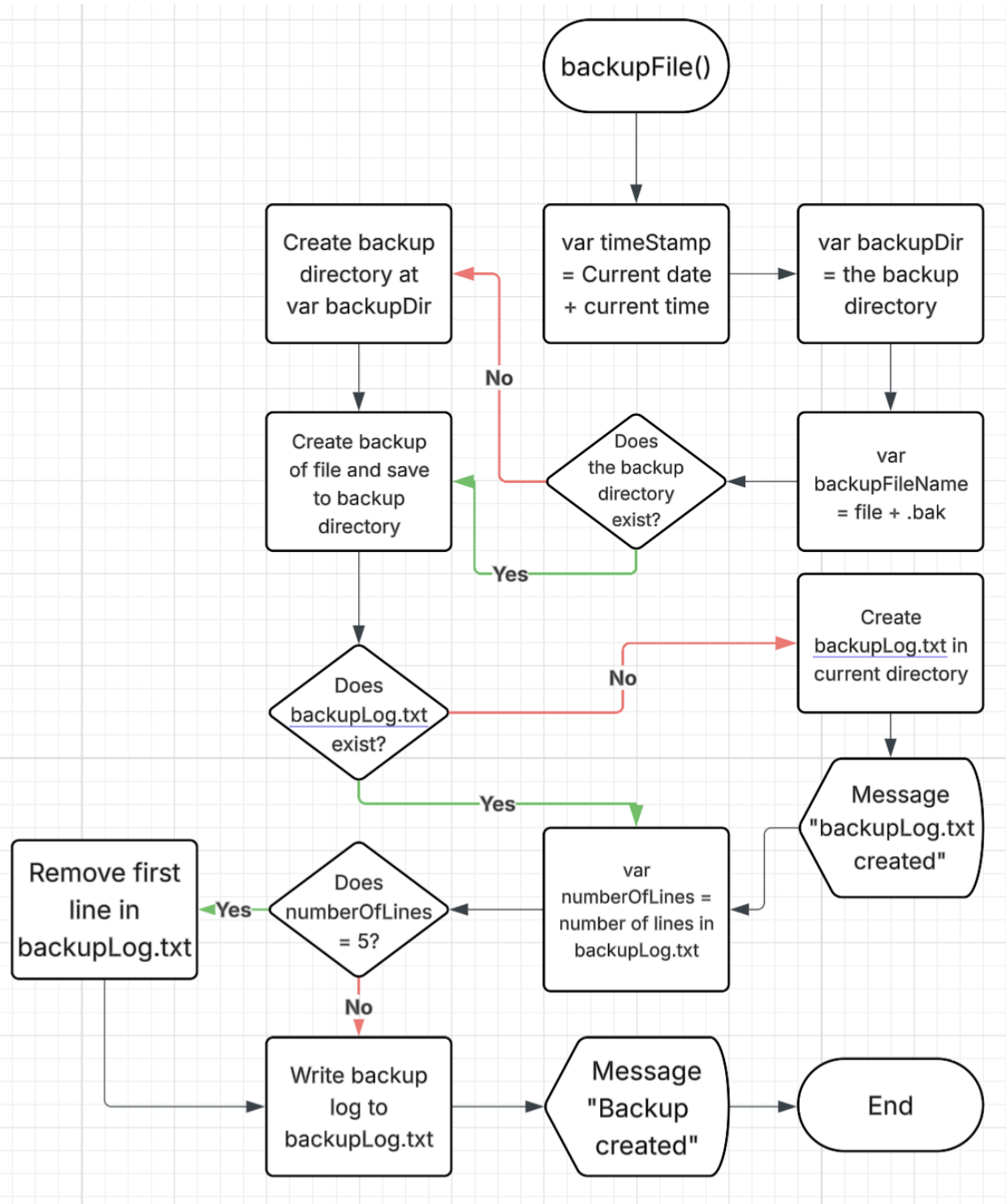
## Flow Charts



The above flow chart shows the general flow of the program without going into detail on how the edit() function works.



This flowchart shows the logic for the `edit()` function that we saw in the first flow chart.



backupFile function flowchart

# Linux Bash Script

## Code

```
#!/bin/bash

# Safe Edit Bash Script

# backup file function
backupFile(){

    # Define filePath var as the file path that was passed to the function
    local filePath="$1"

    # Create timeStamp
    local timestamp=$(date +%Y-%m-%d %H:%M:%S)

    # Define backup dir path
    local backupDir="./backup"

    # Create backup file name by removing file extension and replacing it with
    # .bak
    local backupFileName="${filePath%.*}.bak"

    # Ensure backup directory exists
    if [ ! -d "$backupDir" ]; then

        # Create folder if it doesn't exist
        mkdir -p "$backupDir"
    fi

    # Create backup of file and save to backup folder
    cp "$filePath" "$backupDir/${basename "$filePath"}"

    # Check if cp command was successful
    if [ $? -ne 0 ]; then

        # Show error message to user
    fi
}
```

```

        echo "Error: Failed to create backup for $filePath"

        # Exit program
        exit 0
    fi

    # Check if the backup_log.txt file exists
    if [ ! -f "backup_log.txt" ]; then

        # Create a new backupLogBash if one doesn't exist
        touch backup_log.txt

        # Tell user new backupLogBash has been created
        echo "backup_log.txt created"

        # Give the user a chance to read the echo message
        sleep 2
    fi

    # Check number of lines in backup_log.txt
    local numberOfLines=$(wc -l < backup_log.txt)

    # If backup_log.txt has 5 lines, remove the first one
    if [ "$numberOfLines" -eq 5 ]; then

        # Check platform as the sed command syntax is different on macOS and
linux
        if [[ "$OSTYPE" == "darwin"* ]]; then

            # For macOS
            sed -i '' '1d' backup_log.txt

        else

            # For linux
            sed -i '1d' backup_log.txt
        fi
    fi
fi

```



```
# Write backup log to the file with timestamp
echo "[$timestamp] Backup created: $filePath → $backupFileName" >>
backup_log.txt

# Tell user the backup has been created
echo "Backup of $filePath created successfully at $timestamp"

# Sleep to give the user a chance to read the above echo line
sleep 2
}

# edit function
edit(){

    local filePath="$1"
    # Check whether the filepath exists in the current dir
    if [ -f "$filePath" ]; then

        # Backup the file
        backupFile "$filePath"

        # Open vi editor
        if ! vi "$filePath"; then

            # Show user error message if vi failed to open the file
            echo "Error: Failed to open $filePath with vi."

        fi

    else

        # Give user warning message saying the file doesn't exist
        echo "Error: $filePath does not exist in the current directory."

    fi
}
```

```
# Check if there is a command line argument
if [ $# -eq 1 ]; then

    # User welcome message
    echo "Welcome to the Safe Editor!"
    # If exactly one argument is given, call edit function
    edit "$1"

    # If more than one argument is given
elif [ $# -gt 1 ]; then

    # Display error message too user
    echo "Error: Too many parameters entered."
else

    # User welcome message
    echo "Welcome to the Safe Editor!"

    # Ask user which file they want to edit
    echo "Which file would you like to edit?"

    # Read into var filePath
    read -r filePath

    # Call the edit function with the filePath
    edit "$filePath"
fi
```

## Tests

Note for the purpose of testing “backup\_log” has been renamed to “backupLogBash” for clarity.

---

**Test Number:** 01

**Test Case:** Test with a valid file to edit and back it up

**Input:** test.txt

**Expected Output:** Backup created message: "Backup of testfile.txt created successfully at [timestamp]" and the vi editor opens

**Expected Behaviour:**

- The file should be backed up successfully.
- The vi editor should open.
- The backup log file should be updated with the timestamp.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**

```
localhost:~$ ./safeEdit.sh
Welcome to the Safe Editor!
Which file would you like to edit?
test.txt
```

Prompted to enter a filename as there was no command line argument, so is running in interactive mode

```
localhost:~$ ./safeEdit.sh
Welcome to the Safe Editor!
Which file would you like to edit?
test.txt
Backup of test.txt created successfully at 2025-03-25 20:22:52
```

Confirmation that a backup was created successfully

This is the editor and it is working correctly

This is the vi editor opened correctly

The contents of the backup file showing the timestamp of the backup

---

**Test Number:** 02

**Test Case:** Test with an invalid file (file does not exist)

**Input:** nonexistentfile.txt

**Expected Output:** "Error: nonexistent.txt does not exist in the current directory."

**Expected Behaviour:** The script should output an error message indicating the file does not exist.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**

```
localhost:~$ ./safeEdit.sh
Welcome to the Safe Editor!
Which file would you like to edit?
nonexistentfile.txt
Error: nonexistentfile.txt does not exist in the current directory.
localhost:~$
```

The correct error message being displayed

---

**Test Number:** 03

**Test Case:** Test with 1 argument passed

**Input:** file1.txt

**Expected behaviour:** Backup created successfully message, open vi editor, log backup in file, create backup file

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**

```
localhost:~$ ./safeEdit.sh test.txt
```

Command entered

```
localhost:~$ ./safeEdit.sh test.txt
Welcome to the Safe Editor!
Backup of test.txt created successfully at 2025-03-25 21:01:33
```

Message showing that backup has been successfully created

[illegible]

Vi editor opened successfully

```
localhost:~$ cat backupLogBash.txt
[2025-03-25 20:45:42] Backup created: test.txt → test.bak
[2025-03-25 20:45:49] Backup created: test.txt → test.bak
[2025-03-25 20:47:20] Backup created: test.txt → test.bak
[2025-03-25 20:48:55] Backup created: test2.txt → test2.bak
[2025-03-25 21:01:33] Backup created: test.txt → test.bak
localhost:~$
```

Logged successfully

```
localhost:~/backupBash$ ls
test.bak  test2.bak
localhost:~/backupBash$ cat test.bak
This is the editor and it is working correctly
localhost:~/backupBash$
```

```
.bak file created successfully with the correct contents
```

**Test Number: 04**

### Test Case: Test with more than one argument passed

**Input:** file1.txt file2.txt

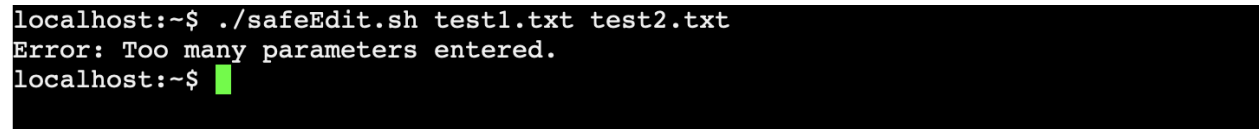
**Expected Output:** "Error: Too many parameters entered."

**Expected Behaviour:** Should output an error message indicating that too many parameters were entered.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**



```
localhost:~$ ./safeEdit.sh test1.txt test2.txt
Error: Too many parameters entered.
localhost:~$
```

A terminal window screenshot with a black background and white text. The first line shows the command './safeEdit.sh test1.txt test2.txt' being executed. The second line shows the output 'Error: Too many parameters entered.'. The third line shows the prompt 'localhost:~\$' followed by a green cursor block.

Image showing the correct error message

---

**Test Number:** 05

**Test Case:** Test backup creation when backupLogBash.txt does not exist

**Input:** test.txt (ensure backup directory doesn't exist)

**Expected Output:** "backupLogBash.txt created", and the backup is successful

**Expected Behaviour:** The script should create the backup directory and the backup log file if they do not exist.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**

```
localhost:~$ ls
backupBash  safeEdit.sh  test.txt
localhost:~$
```

Image showing the contents of the directory at the start, there is no backupLogBash.txt file

```
localhost:~$ ./safeEdit.sh
Welcome to the Safe Editor!
Which file would you like to edit?
test.txt
```

The user input

```
localhost:~$ ./safeEdit.sh
Welcome to the Safe Editor!
Which file would you like to edit?
test.txt
backupLogBash.txt created
```

Message showing that the a new backupLogBash.txt file created

```
localhost:~$ ls
backupBash  backupLogBash.txt  safeEdit.sh  test.txt
localhost:~$
```

The directory afterwards, showing the new backupLogBash.txt file

```
localhost:~$ cat backupLogBash.txt
[2025-03-25 20:40:29] Backup created: test.txt → test.bak
localhost:~$
```

The contents of the file showing that it has been created correctly and contains the backup timestamp



---

**Test Number:** 06

**Test Case:** Test backup when the `backupLogBash.txt` file has 5 lines

**Input:** `test2.txt` and 5 entries in `backupLogBash.txt`

**Expected Output:** The first line in `backupLogBash.txt` is deleted, and a new entry is added.

**Expected Behaviour:** When the backup log reaches 5 lines, the oldest entry should be removed and the new one added.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**

```
localhost:~$ cat backupLogBash.txt
[2025-03-25 20:45:29] Backup created: test.txt → test.bak
[2025-03-25 20:45:35] Backup created: test.txt → test.bak
[2025-03-25 20:45:42] Backup created: test.txt → test.bak
[2025-03-25 20:45:49] Backup created: test.txt → test.bak
[2025-03-25 20:47:20] Backup created: test.txt → test.bak
localhost:~$
```

`backupLogBash.txt` before the test with 5 lines

```
localhost:~$ cat backupLogBash.txt
[2025-03-25 20:45:35] Backup created: test.txt → test.bak
[2025-03-25 20:45:42] Backup created: test.txt → test.bak
[2025-03-25 20:45:49] Backup created: test.txt → test.bak
[2025-03-25 20:47:20] Backup created: test.txt → test.bak
[2025-03-25 20:48:55] Backup created: test2.txt → test2.bak
localhost:~$
```

After showing the new entry at the the bottom of the list and the first entry removed keeping the total number of lines at five

---

# Windows Batch Script

## Code

```
@echo off
setlocal enabledelayedexpansion

:: Start of by calling the main function
goto :main

:: backup file function
:backupFile

:: Define filePath var as the file path that was passed to the function
set "filePath=%~1"

:: Create timeStamp with yyyy-mm-dd h:mm:ss format
for /f "tokens=1-3 delims=/" %%a in ('echo %date%') do (
    set "day=%%a"
    set "month=%%b"
    set "year=%%c"
)

:: Remove any leading spaces
set "day=%day: =%"

:: Get time with seconds
for /f "tokens=1-3 delims=:." %%a in ('echo %time%') do (
    set "hour=%%a"
    set "minute=%%b"
    set "second=%%c"
)

:: Remove any leading spaces
set "hour=%hour: =%"

:: Only keep first two digits of seconds
set "second=%second:~0,2%"
```

```

:: Format as yyyy-mm-dd h:mm:ss
set "timestamp=%year%-month%-day% %hour%:%minute%:%second%"

:: backup dir path
set "backupDir=.\backup"

:: Create backup file name by removing file extension and replacing it with
.bak
for %%i in ("%filePath%") do set "backupFileName=%%~ni.bak"

:: Ensure backup directory exists
if not exist "%backupDir%" (

    :: Create folder if it doesn't exist
    mkdir "%backupDir%"
)

:: Create backup of file and save to backup folder
copy "%filePath%" "%backupDir%\%backupFileName%" >nul

:: Check if copy command was successful
if errorlevel 1 (

    :: Show error message to user
    echo Error: Failed to create backup for %filePath%

    :: Exit function
    goto :eof
)

:: Check if the backup_log.txt file exists
if not exist "backup_log.txt" (

    :: Create a new backupLogBatch if one doesn't exist
    type nul > backup_log.txt

    :: Tell user new backupLogBatch has been created

```

```

    echo backup_log.txt created

    :: Give the user a chance to read the echo message
    timeout /t 2 >nul
)

:: Check number of lines in backup_log.txt
for /f %%a in ('type "backup_log.txt" ^| find /c /v "') do set
"numberOfLines=%%a"

:: If backup_log.txt has 5 lines, remove the first one
if "%numberOfLines%"=="5" (

    :: Create a temporary file without the first line
    more +1 "backup_log.txt" > "backup_log.tmp"

    :: Replace the original file with the temp file
    move /y "backup_log.tmp" "backup_log.txt" >nul
)

:: Write backup log to the file with timestamp
echo [%timestamp%] Backup created: %filePath% -^> %backupFileName% >>
backup_log.txt

:: Tell user the backup has been created
echo Backup of %filePath% created successfully at %timestamp%

:: Sleep to give the user a chance to read the above echo line
timeout /t 2 >nul

goto :eof

:: edit function
:edit

set "filePath=%~1"

:: Check whether the filepath exists in the current dir

```

```

if exist "%filePath%" (

    :: Backup the file
    call :backupFile "%filePath%"

    :: Open notepad editor
    start /wait notepad "%filePath%"

    if errorlevel 1 (
        :: Show user error message if notepad failed to open the file

        echo Error: Failed to open %filePath% with notepad.
    )
) else (

    :: Give user warning message saying the file doesn't exist
    echo Error: %filePath% does not exist in the current directory.
)

goto :eof

:main
:: Check if there is a command line argument
if "%~1" neq "" (
    if "%~2" equ "" (

        :: User welcome message
        echo Welcome to the Safe Editor!

        :: If exactly one argument is given, call edit function
        call :edit "%~1"

    ) else (
        :: If more than one argument is given
        :: Display error message to user
        echo Error: Too many parameters entered.
    )
)

```

```

) else (

  :: User welcome message
  echo Welcome to the Safe Editor!

  :: Ask user which file they want to edit
  echo.

  echo Which file would you like to edit?

  :: Read into var filePath
  set /p filePath=

  :: Call the edit function with the filePath
  call :edit "!filePath!"
)

:end

endlocal

```

## Tests

Note for the purpose of testing “backup\_log” has been renamed to “backupLogBatch” for clarity.

---

### Test Number: 01

**Test Case:** Test with a valid file to edit and back it up

**Input:** test.txt

**Expected Output:** Backup created message: "Backup of testfile.txt created successfully at [timestamp]" and the notepad editor opens

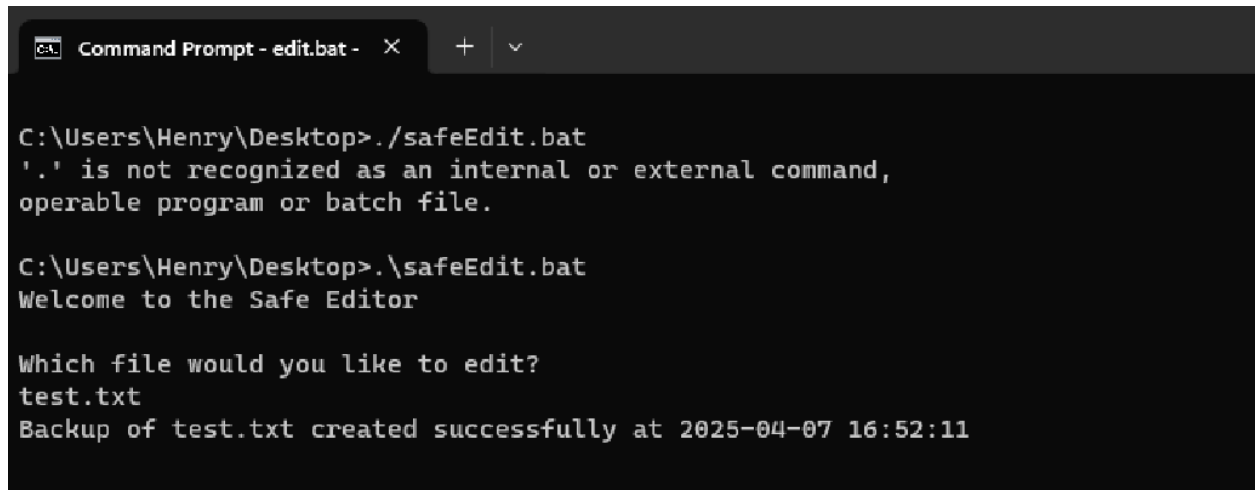
**Expected Behaviour:**

- The file should be backed up successfully.
- The notepad editor should open.
- The backup log file should be updated with the timestamp.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**



```
Command Prompt - edit.bat - X + v

C:\Users\Henry\Desktop>./safeEdit.bat
'.' is not recognized as an internal or external command,
operable program or batch file.

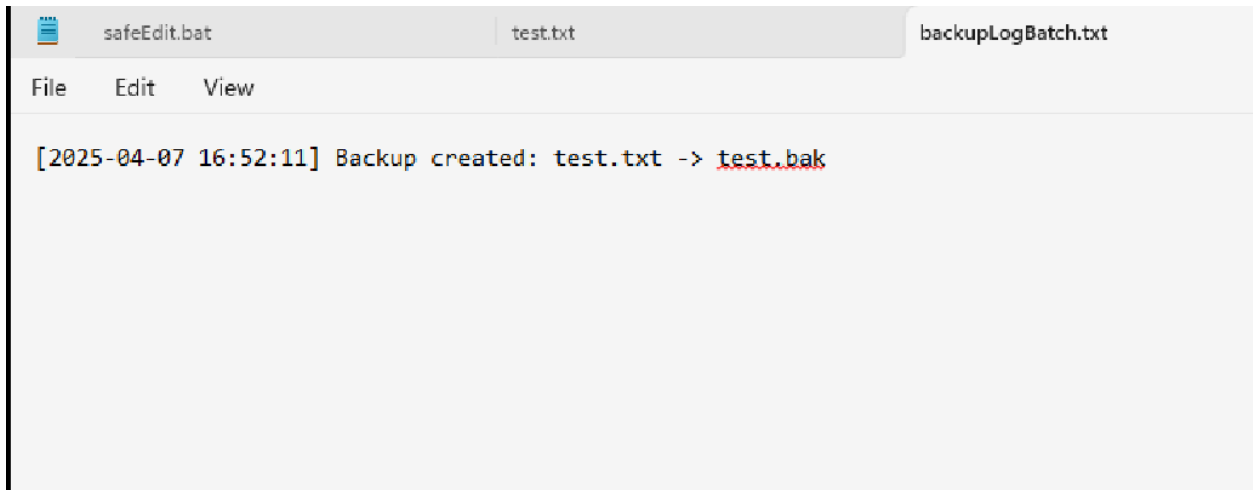
C:\Users\Henry\Desktop>.\safeEdit.bat
Welcome to the Safe Editor

Which file would you like to edit?
test.txt
Backup of test.txt created successfully at 2025-04-07 16:52:11
```

Message showing backup has been created successfully.



The file opened successfully in notepad



The added contents to the backup log file

```
C:\Users\Henry\Desktop\backupBatch>dir
Volume in drive C has no label.
Volume Serial Number is C67B-B102

Directory of C:\Users\Henry\Desktop\backupBatch

07/04/2025  16:26    <DIR>          .
07/04/2025  16:51    <DIR>          ..
07/04/2025  16:26                99 test.bak
               1 File(s)                99 bytes
               2 Dir(s)  32,657,502,208 bytes free

C:\Users\Henry\Desktop\backupBatch>
```

Finally the new .bak file



---

**Test Number:** 02

**Test Case:** Test with an invalid file (file does not exist)

**Input:** nonexistentfile.txt

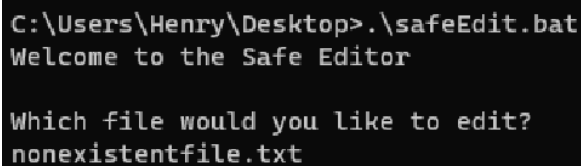
**Expected Output:** "Error: nonexistentfile.txt does not exist in the current directory."

**Expected Behaviour:** The script should output an error message indicating the file does not exist.

**Actual Output:** As expected

**Actual Behaviour:** As expected

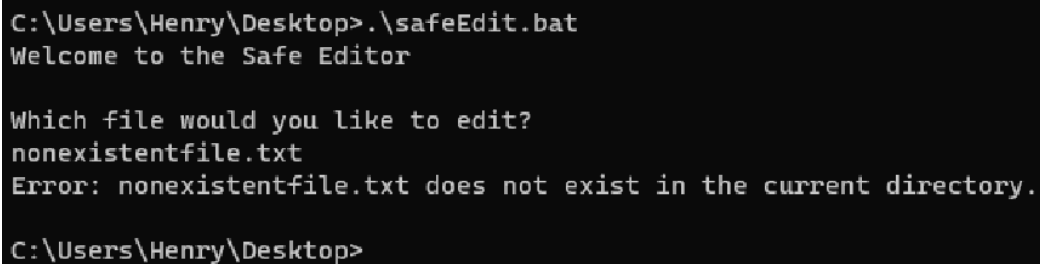
**Images:**



```
C:\Users\Henry\Desktop>.\safeEdit.bat
Welcome to the Safe Editor

Which file would you like to edit?
nonexistentfile.txt
```

The input



```
C:\Users\Henry\Desktop>.\safeEdit.bat
Welcome to the Safe Editor

Which file would you like to edit?
nonexistentfile.txt
Error: nonexistentfile.txt does not exist in the current directory.

C:\Users\Henry\Desktop>
```

And the correct error message

---

**Test Number:** 03

**Test Case:** Test with 1 argument passed

**Input:** file1.txt

**Expected behaviour:** Backup created successfully message, open notepad editor, log backup in file, create backup file

**Actual Output:** As expected

**Actual Behaviour:** As expected

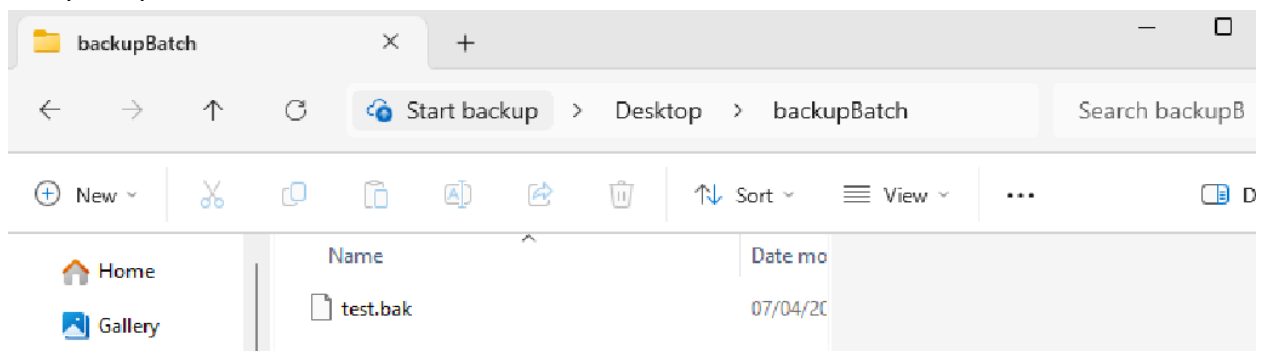
**Images:**

```
C:\Users\Henry\Desktop>.\safeEdit.bat test.txt
Welcome to the Safe Editor
Backup of test.txt created successfully at 2025-04-07 17:08:35
```

Exempted argument and created backup



Notepad opened



.bak file created

```
[2025-04-07 16:49:45] Backup created: test.txt -> test.bak  
[2025-04-07 16:52:11] Backup created: test.txt -> test.bak  
[2025-04-07 17:08:35] Backup created: test.txt -> test.bak  
[2025-04-07 17:09:21] Backup created: test.txt -> test.bak
```

I

And backup message logged

---

**Test Number:** 04

**Test Case:** Test with more than one argument passed

**Input:** file1.txt file2.txt

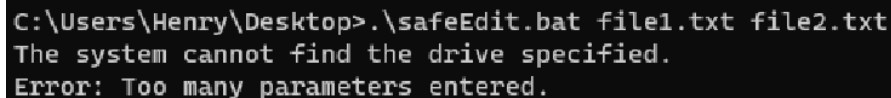
**Expected Output:** "Error: Too many parameters entered."

**Expected Behaviour:** Should output an error message indicating that too many parameters were entered.

**Actual Output:** As expected

**Actual Behaviour:** As expected

**Images:**



```
C:\Users\Henry\Desktop>.\safeEdit.bat file1.txt file2.txt  
The system cannot find the drive specified.  
Error: Too many parameters entered.
```

Image showing the correct error message

---

**Test Number:** 05

**Test Case:** Test backup creation when backupLogBatch.txt does not exist

**Input:** test.txt (ensure backup directory doesn't exist)

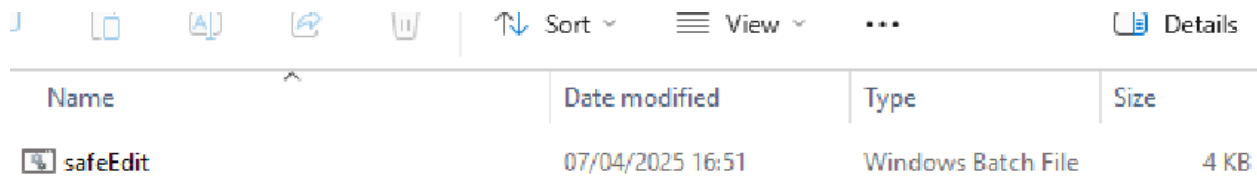
**Expected Output:** "backupLogBatch.txt created", and the backup is successful


**Expected Behaviour:** The script should create the backup directory and the backup log file if they do not exist.

**Actual Output:** As expected

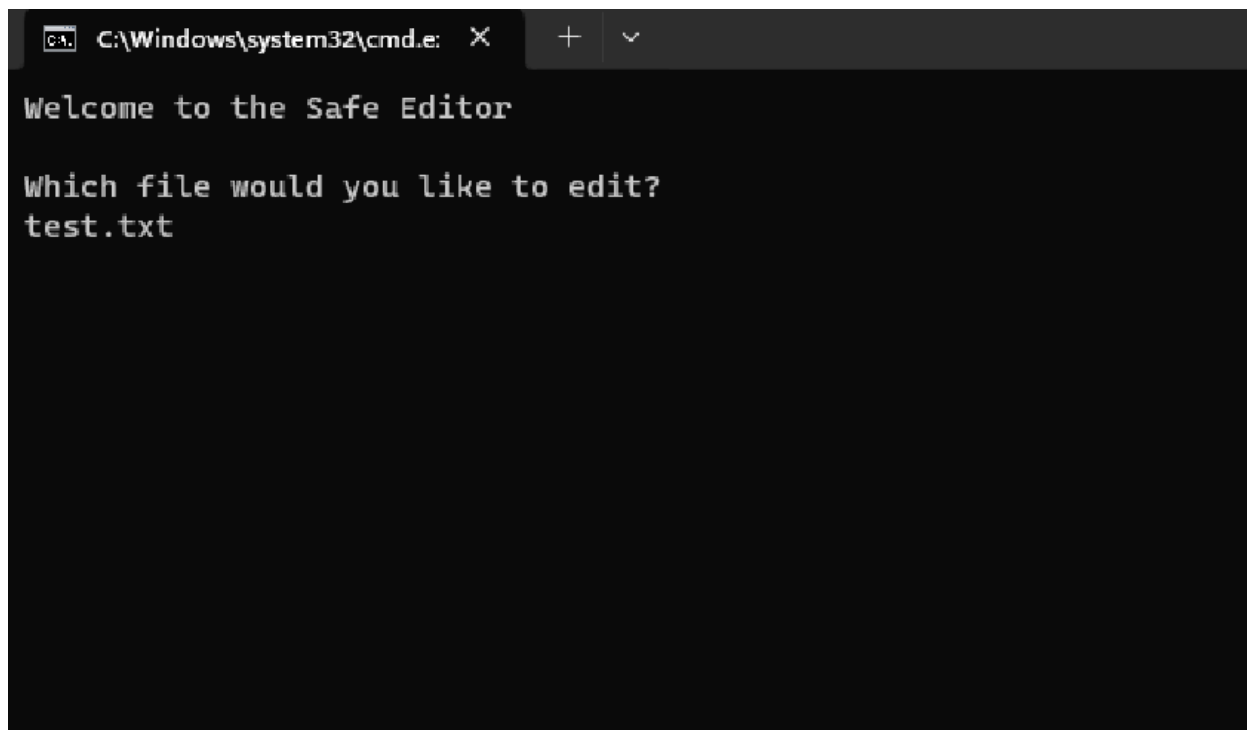
**Actual Behaviour:** As expected

**Images:**



Name	Date modified	Type	Size
 safeEdit	07/04/2025 16:51	Windows Batch File	4 KB

No backupLogBatch.txt present

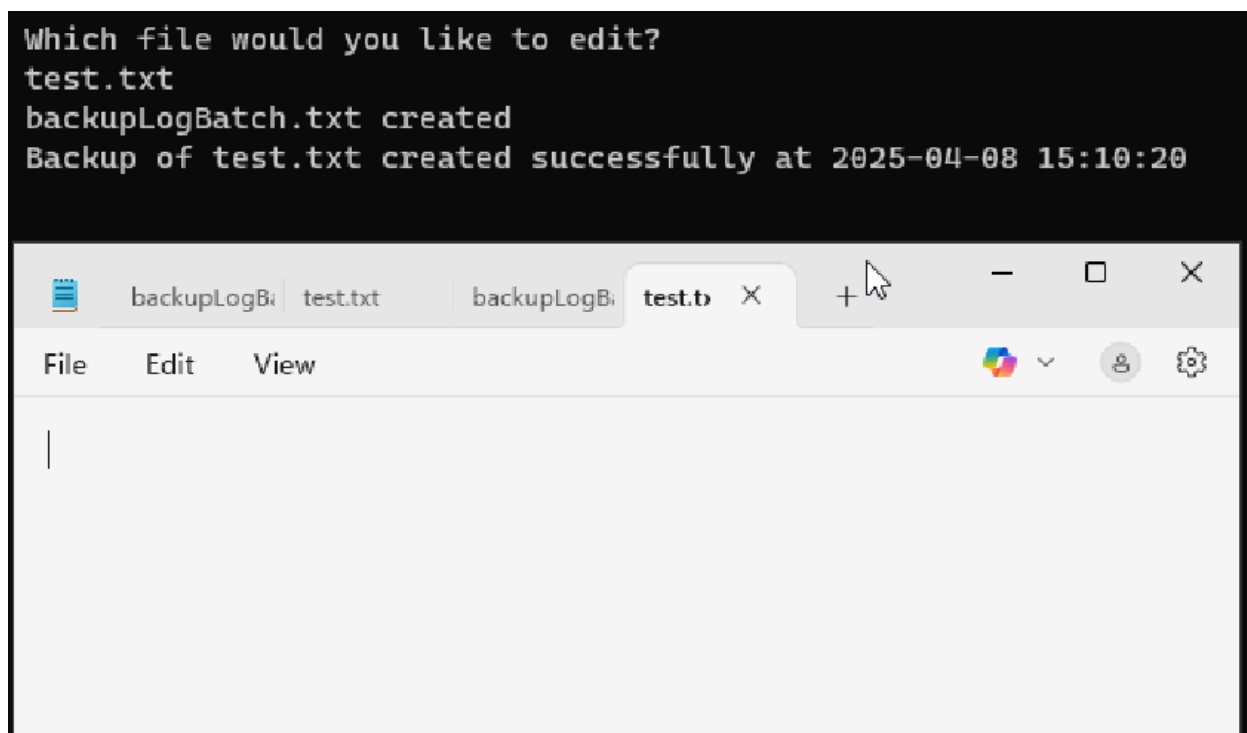


```
C:\Windows\system32\cmd.exe X + v

Welcome to the Safe Editor





Which file would you like to edit?
test.txt
```

Input



```
Which file would you like to edit?
test.txt
backupLogBatch.txt created
Backup of test.txt created successfully at 2025-04-08 15:10:20
```

Backup log created, backup logged and notepad opened

name	Date modified	type	Size
 safeEdit	07/04/2025 16:51	Windows Batch File	4 K
 test	08/04/2025 15:10	Text Document	0 K
 backupBatch	08/04/2025 15:10	File folder	
 backupLogBatch	08/04/2025 15:10	Text Document	1 K



Confirmation that the backupLogBatch.txt file has been created

---

**Test Number:** 06

**Test Case:** Test backup when the backupLogBatch.txt file has 5 lines

**Input:** test.txt and 5 entries in backupLogBatch.txt

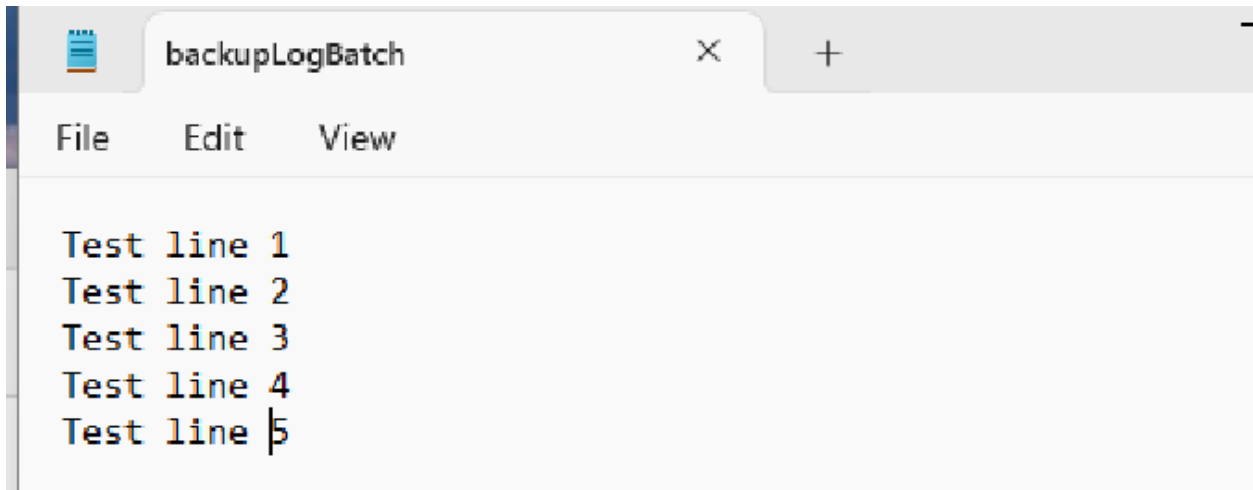
**Expected Output:** The first line in backupLogBatch.txt is deleted, and a new entry is added.

**Expected Behaviour:** When the backup log reaches 5 lines, the oldest entry should be removed and the new one added.

**Actual Output:** As expected

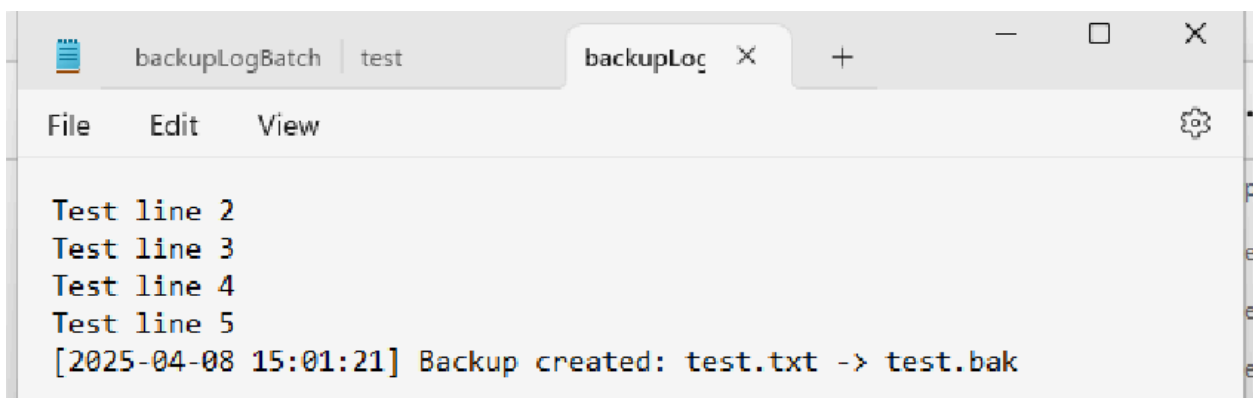
**Actual Behaviour:** As expected

**Images:**



```
Test line 1
Test line 2
Test line 3
Test line 4
Test line 5
```

Starting contents of backuplogbatch.txt



```
Test line 2
Test line 3
Test line 4
Test line 5
[2025-04-08 15:01:21] Backup created: test.txt -> test.bak
```

And the contents after running the program with the new backup log and the oldest log (Test line 1) removed

---

