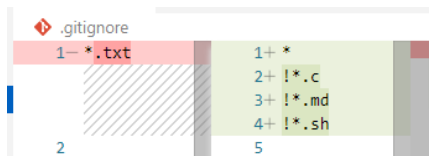


## Exercise 2

Use bash and git commands

1. Make a directory, **exercise-2**, on the desktop of your computer and open it in visual studio code.
2. Open the terminal of visual studio code and create a repository and run **git status** and **git log**
3. Create **README.md** file of your repository and write **# Exercise 2** to it
4. Create a **.gitignore** file and ignore all **.txt** files.
5. Add the changes to the **staging** area of the repository.
6. Make a **commit** with message "Initial commit"
7. Run **git status**, **git log** and **git log --oneline**.

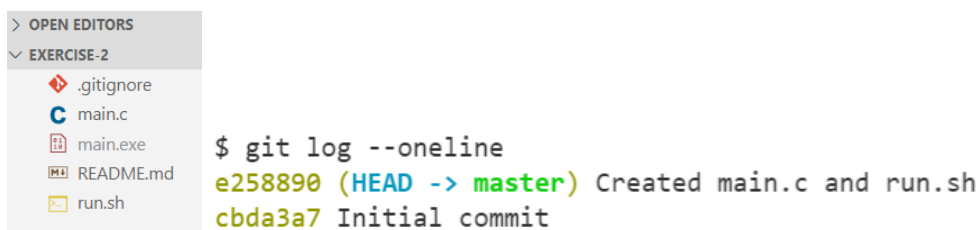
```
$ git log --oneline
02a0a7f (HEAD -> master) Initial commit
```
8. Remove **README.md** from the repository using **git rm**.
9. Run **git status** and then **unstage** the change using **git restore**.
10. Run **git status** and **discard** changes using **git restore**.
11. Instead of all **.txt** files, ignore all files whose extensions are not **.c**, **.md** and **.sh**



12. Run **git status** and add **.gitignore** to the **staging** area.
13. Create two files, **main.c** and **run.sh**, in the root of the repo.
14. Write the below code using **printf** to **run.sh**

```
clear && gcc main.c -o main && ./main
```
15. Write the below code using **printf** to **main.c**

```
#include <stdio.h>\n\nint main(void) {\n    printf("Hello, World!");\n    return 0;\n}
```
16. Run **git status** and add the changes to the staging area
17. **Commit** changes with message "Created main.c and run.sh"
18. Run **sh run.sh** in the terminal. Has the executable file, **main.exe** or **main**, been ignored?



19. Run **git log**. Change the message of the last commit to "First commit"
20. Add a **note**, *The program and its compilation*, to the last commit.
21. Create a **branch**, *feature-branch*, based on the *master branch*
22. Create another **branch**, *print-1-6-1*, based on the *master branch*
23. Get the list of branches using **git branch**
24. **Rename** the *feature-branch* branch to **print-1-3-1**

25. Get the list of branches using **git branch**

```
$ git branch
* master
  print-1-3-1
  print-1-6-1
```

26. **Switch** to *print-1-3-1* branch and run **git log --oneline**

27. In **main.c** make a program using a **for loop** to **print from 1 to 3** to the **output**

```
C main.c > ...
1  #include <stdio.h>
2
3  int main(void)
4  {
5      for (int i = 1; i < 4; i++)
6      {
7          printf("%d ", i);
8      }
9      printf("\n");
10
11     return 0;
12 }
```

28. Run **sh run.sh** in the **terminal** and be sure your program works.

29. Run **git status** and **add** the changes to the **staging** area.

30. **Commit** the changes with the message "**print from 1 to 3**". Run **git log --oneline**.

```
$ git log --oneline
e92faea (HEAD -> print-1-3-1) print from 1 to 3
dc6934a (print-1-6-1, master) First commit
02a0a7f Initial commit
```

31. Now **switch to master** and run **git log --oneline**.

32. What is the **difference** between *master* and *print-1-3-1*?

33. **Switch** to *print-1-3-1* and run **git status**.

34. In **main.c** make a **for loop** after the previous loop to **print from 2 to 1** to the **output**

```
C main.c > ...
1  #include <stdio.h>
2
3  int main(void)
4  {
5      for (int i = 1; i < 4; i++)
6      {
7          printf("%d ", i);
8      }
9
10     for (int i = 2; i > 0; i--)
11     {
12         printf("%d ", i);
13     }
14
15     printf("\n");
16
17     return 0;
18 }
```

35. Run **sh run.sh** in the **terminal** and be sure your program works.

36. **Commit** the changes with message “**print from 2 to 1**” and run **git log --oneline**

```
$ git log --oneline
147d7f1 (HEAD -> print-1-3-1) print from 2 to 1
e92faea print from 1 to 3
dc6934a (print-1-6-1, master) First commit
02a0a7f Initial commit
```

37. In the last loop change your code in order to print from **12 to 1** to the output.
38. Run **sh run.sh** in the **terminal** and be sure your program works.
39. **Commit** the changes with message “**print from 12 to 1**” and run **git log --oneline**
40. **Revert** the **last commit** with the message “**Revert print from 12 to 1**”. Run **git log --oneline**

```
$ git log --oneline
38a8cf9 (HEAD -> print-1-3-1) Revert "print from 12 to 1"
db1ab4f print from 12 to 1
147d7f1 print from 2 to 1
e92faea print from 1 to 3
dc6934a (print-1-6-1, master) First commit
02a0a7f Initial commit
```

41. Then **hard reset** the branch to the commit with message “**print from 2 to 1**”
42. **Merge** *print-1-3-1* branch into master with the message “**print from 1 to 3 to 1**”
43. Run **git log --oneline**. Delete branch *print-1-3-1* and run **git branch**

```
$ git log --oneline
147d7f1 (HEAD -> master) print from 2 to 1
e92faea print from 1 to 3
dc6934a (print-1-6-1) First commit
02a0a7f Initial commit

$ git branch
* master
  print-1-6-1
```

44. **Switch** to *print-1-6-1* branch and run **git log --oneline**
45. Use **git cherry-pick** and add the commit with message “**print from 12 to 1**” to the branch

```
C main.c > main(void)
1  #include <stdio.h>
2
3  int main(void)
4  {
5      Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
6      <<<<<< HEAD (Current Change)
7      =====
8      for (int i = 1; i < 4; i++)
9      {
10         printf("%d ", i);
11     }
12     for (int i = 12; i > 0; i--)
13     {
14         printf("%d ", i);
15     }
16     printf("\n");
17
18     >>>>>> db1ab4f (print from 12 to 1) (Incoming Change)
19     return 0;
20
21 }
```

46. Is there a conflict? solve it in a way that the program counts from 1 to 6 and then 5 to 1.
47. Run **git status** and **add** the changes to the **staging** area
48. **Commit** changes with message “**print from 1 to 6 to 1**”
49. Add a comment, *// Print from 1 to 6 to the output*, to the first loop in main.c

50. Add a comment, **// Print from 5 to 1 to the output**, to the second loop in main.c

```
C main.c > ...
1  #include <stdio.h>
2
3  int main(void)
4  {
5      // Print from 1 to 6 to the output
6      for (int i = 1; i < 7; i++)
7      {
8          printf("%d ", i);
9      }
10
11     // Print from 5 to 1 to the output
12     for (int i = 5; i > 0; i--)
13     {
14         printf("%d ", i);
15     }
16
17     printf("\n");
18
19     return 0;
20 }
```

51. Try to switch to master. Is it possible? Use **git stash** to save changes and then switch to master.  
52. Run **git log --oneline** and then **switch** to *print-1-6-1*  
53. Use **git stash list** to get the list of stashes. Then restore the stash using **git stash pop**  
54. Add changes to the **staging** area and then **commit** changes with message **"Commented the code"**  
55. Merge *print-1-6-1* into *master* with message **"count and print 1-6-1"**.  
56. Is there a conflict? solve it and use **git merge --continue** to complete the merge. Run **git log --oneline**.

```
$ git log --oneline
5c3f9ab (HEAD -> master) count and print 1-6-1
453d2c6 (print-1-6-1) commented the code
124c478 print from 1 to 6 to 1
147d7f1 print from 2 to 1
e92faea print from 1 to 3
dc6934a First commit
02a0a7f Initial commit
```

57. Delete *print-1-6-1* and run **git branch** and **git log --decorate --graph --oneline**

```
$ git log --decorate --graph --oneline
* 5c3f9ab (HEAD -> master) count and print 1-6-1
| \
| * 453d2c6 commented the code
| * 124c478 print from 1 to 6 to 1
| * 147d7f1 print from 2 to 1
| * e92faea print from 1 to 3
|/
* dc6934a First commit
* 02a0a7f Initial commit
```

58. Add a **tag, v1.0**, to the **last commit** and run **git tag** to list the tags  
59. Run **git log**, **git log --oneline** and then add a message, **The first release**, to the tag  
60. Run **git tag** and **git tag -n** to show the tag and then **delete** the tag.

```
$ git tag -n
v1.0          The first release
```