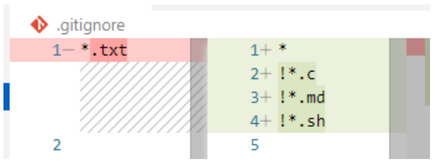
Exercise 2

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
c829308 (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**



```
.gitignore
- *.txt
+ *
+ !*.c
+ !*.md
+ !*.sh
```

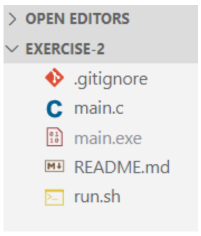
12. Run **git status** and add **.gitignore** to the **staging** area.
13. Create two files, **main.c** and **run.sh,** in the root of the repo.
14. Write the below code using **printf** to **run.sh**

```
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{\n\treturn 0;\n}
```

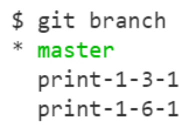
16. Run **git status** and add the changes to the staging area.
17. **Commit** changes with message "**First commit**"
18. Run **sh run.sh** in the terminal. Has the executable file, **main.exe** or **main**, been ignored?



```
$ git log --oneline
d40237c (HEAD -> master) First commit
c829308 Initial commit
```

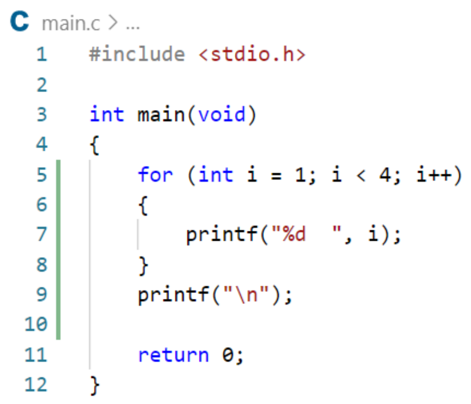
19. Run **git log**. Change the message of the last commit to "**Created main.c and run.sh**"
20. Add a **note**, *The program and its compilation*, to the last commit.
21. Create a **branch**, *feature-branch*, based on the *master branch.*
22. Create another **branch**, *print-1-6-1*, based on the *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**



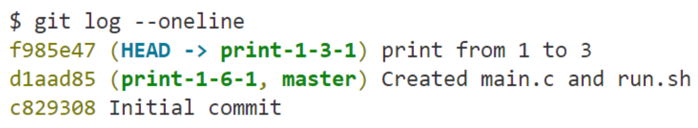
```
$ git branch
* master
  print-1-3-1
  print-1-6-1
```

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**



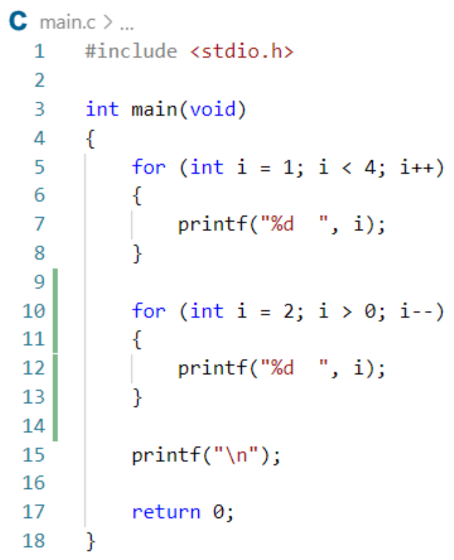
```c
main.c > ...
#include <stdio.h>

int main(void)
{
    for (int i = 1; i < 4; i++)
    {
        printf("%d  ", i);
    }
    printf("\n");

    return 0;
}
```

28. Run **sh run.sh** in the **terminal** and ensure that 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
f985e47 (HEAD -> print-1-3-1) print from 1 to 3
d1aad85 (print-1-6-1, master) Created main.c and run.sh
c829308 Initial commit
```

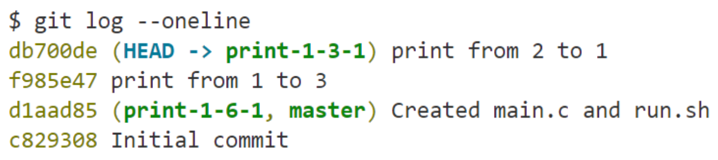
31. Now **switch to master** and run **git log --oneline**.
32. What is the **difference** between ***master*** and ***print-1-3-1****?*
33. **Switch** to ***print-1-3-1*** and run **git status**.
34. In **main.c** make a **for loop** after the previous loop to **print from 2 to 1** to the **output**



```c
main.c > ...
#include <stdio.h>

int main(void)
{
    for (int i = 1; i < 4; i++)
    {
        printf("%d  ", i);
    }

    for (int i = 2; i > 0; i--)
    {
        printf("%d  ", i);
    }

    printf("\n");

    return 0;
}
```

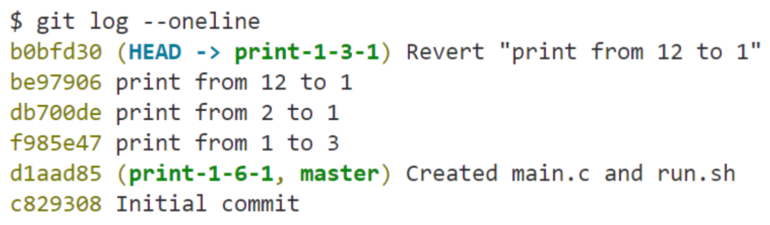
35. Run **sh run.sh** in the **terminal** and be sure your program works.

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



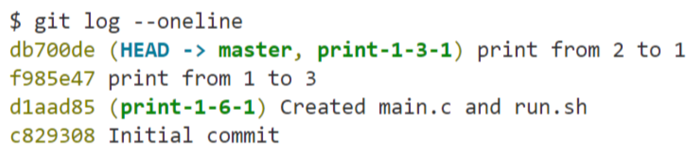
```
$ git log --oneline
db700de (HEAD -> print-1-3-1) print from 2 to 1
f985e47 print from 1 to 3
d1aad85 (print-1-6-1, master) Created main.c and run.sh
c829308 Initial commit
```

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**

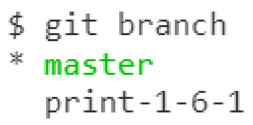


```
$ git log --oneline
b0bfd30 (HEAD -> print-1-3-1) Revert "print from 12 to 1"
be97906 print from 12 to 1
db700de print from 2 to 1
f985e47 print from 1 to 3
d1aad85 (print-1-6-1, master) Created main.c and run.sh
c829308 Initial commit
```

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**

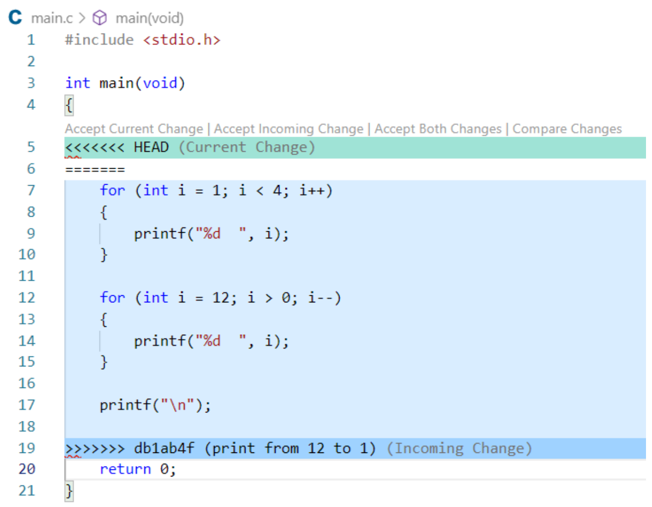


```
$ git log --oneline
db700de (HEAD -> master, print-1-3-1) print from 2 to 1
f985e47 print from 1 to 3
d1aad85 (print-1-6-1) Created main.c and run.sh
c829308 Initial commit
```



```
$ 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

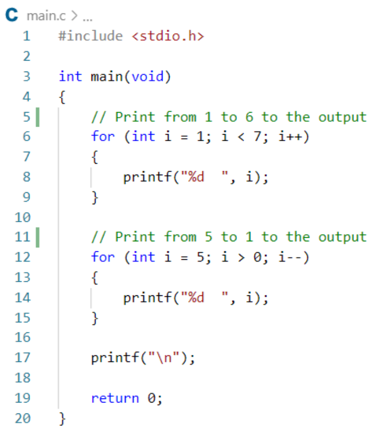


C main.c > main(void)
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
#include <stdio.h>

int main(void)
{
<<<<<<< HEAD (Current Change)
=======
    for (int i = 1; i < 4; i++)
    {
        printf("%d  ", i);
    }

    for (int i = 12; i > 0; i--)
    {
        printf("%d  ", i);
    }

    printf("\n");

>>>>>>> db1ab4f (print from 12 to 1) (Incoming Change)
    return 0;
}
```

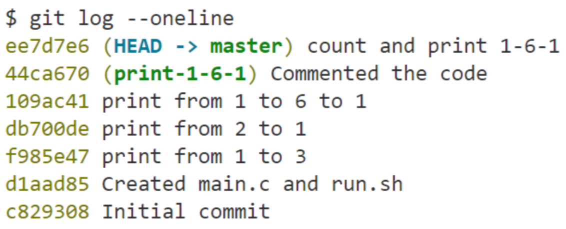
46. Is there a conflict? solve it in a way that the program counts from 1 to 6 and then 5 to 1.
47. Run **git status** and **add** the changes to the **staging** area
48. **Commit** changes with message “**print from 1 to 6 to 1**”
49. Add a comment, ***// Print from 1 to 6 to the output***, to the first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

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



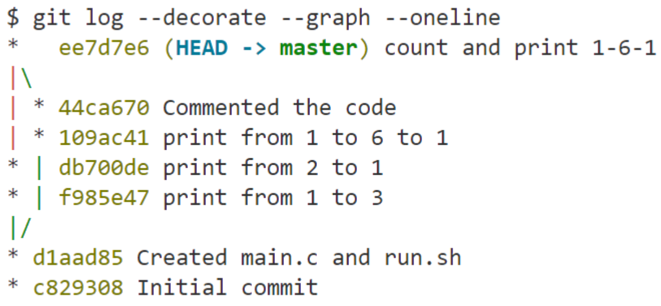
```c
#include <stdio.h>

int main(void)
{
    // Print from 1 to 6 to the output
    for (int i = 1; i < 7; i++)
    {
        printf("%d  ", i);
    }

    // Print from 5 to 1 to the output
    for (int i = 5; i > 0; i--)
    {
        printf("%d  ", i);
    }

    printf("\n");

    return 0;
}
```

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**.



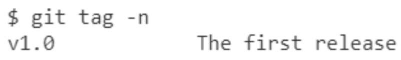
```
$ git log --oneline
ee7d7e6 (HEAD -> master) count and print 1-6-1
44ca670 (print-1-6-1) Commented the code
109ac41 print from 1 to 6 to 1
db700de print from 2 to 1
f985e47 print from 1 to 3
d1aad85 Created main.c and run.sh
c829308 Initial commit
```

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



```
$ git log --decorate --graph --oneline
*   ee7d7e6 (HEAD -> master) count and print 1-6-1
|\
| * 44ca670 Commented the code
| * 109ac41 print from 1 to 6 to 1
* | db700de print from 2 to 1
* | f985e47 print from 1 to 3
|/
* d1aad85 Created main.c and run.sh
* c829308 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
```