Anirudh R
Reg. No.: 210905318

LAB 5

GDB)

```
student@selab-01:~/Desktop/210905318/OSTL/L5$ mkdir gdb
student@selab-01:~/Desktop/210905318/OSTL/L5$ cd gdb/
student@selab-01:~/Desktop/210905318/OSTL/L5/gdb$ touch t.c
student@selab-01:~/Desktop/210905318/OSTL/L5/gdb$ nano t.c
student@selab-01:~/Desktop/210905318/OSTL/L5/gdb$ cat t.c
#include<stdio.h>
int main(){
int num;
do{
printf("Enter a positive integer");
scanf("%d",&num);
} while(num<0);
int factorial, i;
for (i = 1; i<=num; i++){
factorial = factorial *i;
}
printf("%d! = %d\n", num, factorial);
return 0;
}
student@selab-01:~/Desktop/210905318/OSTL/L5/gdb$ gcc -g -o main t.c
```

```
student@selab-01:~/Desktop/210905318/OSTL/L5/gdb$ gdb main
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from main...
(gdb) run
Starting program: /home/student/Desktop/210905318/OSTL/L5/gdb/main
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter a positive integer4
4! = 0
[Inferior 1 (process 2845) exited normally]
(gdb) break main
Breakpoint 1 at 0x555555555195: file t.c, line 2.
(gdb) run
Starting program: /home/student/Desktop/210905318/OSTL/L5/gdb/main
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
```

```
Breakpoint 1, main () at t.c:2
2       int main(){
(gdb) next
5       printf("Enter a positive integer");
(gdb) next
6       scanf("%d",&num);
(gdb) next
Enter a positive integer4
7       } while(num<0);
(gdb) list
2       int main(){
3       int num;
4       do{
5       printf("Enter a positive integer");
6       scanf("%d",&num);
7       } while(num<0);
8       int factorial, i;
9       for (i = 1; i<=num; i++){
10      factorial = factorial *i;
11      }
(gdb) list
12      printf("%d! = %d\n", num, factorial);
13      return 0;
14      }
(gdb) next
9       for (i = 1; i<=num; i++){
(gdb) print num
$1 = 4
(gdb) next
10      factorial = factorial *i;
```

```
$1 = 4
(gdb) next
10      factorial = factorial *i;
(gdb) next
9       for (i = 1; i<=num; i++){
(gdb) next
10      factorial = factorial *i;
(gdb) next
9       for (i = 1; i<=num; i++){
(gdb) next
10      factorial = factorial *i;
(gdb) next
9       for (i = 1; i<=num; i++){
(gdb) next
10      factorial = factorial *i;
(gdb) next
9       for (i = 1; i<=num; i++){
(gdb) next
12      printf("%d! = %d\n", num, factorial);
(gdb) next
4! = 0
13      return 0;
(gdb) print factorial
$2 = 0
(gdb) break t.c:8
Breakpoint 2 at 0x5555555551da: file t.c, line 9.
(gdb) next
14      }
(gdb) print i
$3 = 5
```

```
(gdb) run
Starting program: /home/student/Desktop/210905318/OSTL/L5/gdb/main
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter a positive integer4
4! = 24
[Inferior 1 (process 5761) exited normally]
```

GIT)

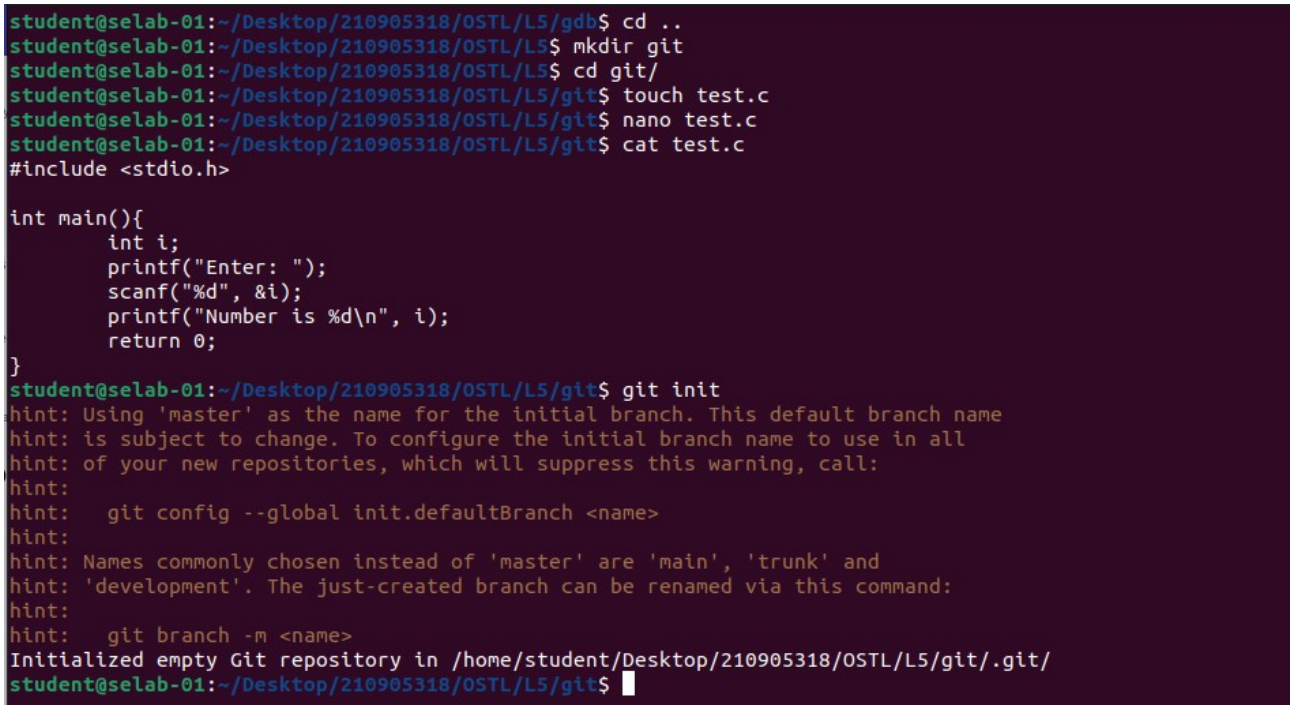
1)Write a simple C program (HelloWorld.c) that takes a number as an input and print the same number as an output. Now create a new git repository (check the contents of .git directory).

```
#include <stdio.h>

int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
        printf("Number is %d\n", i);
```

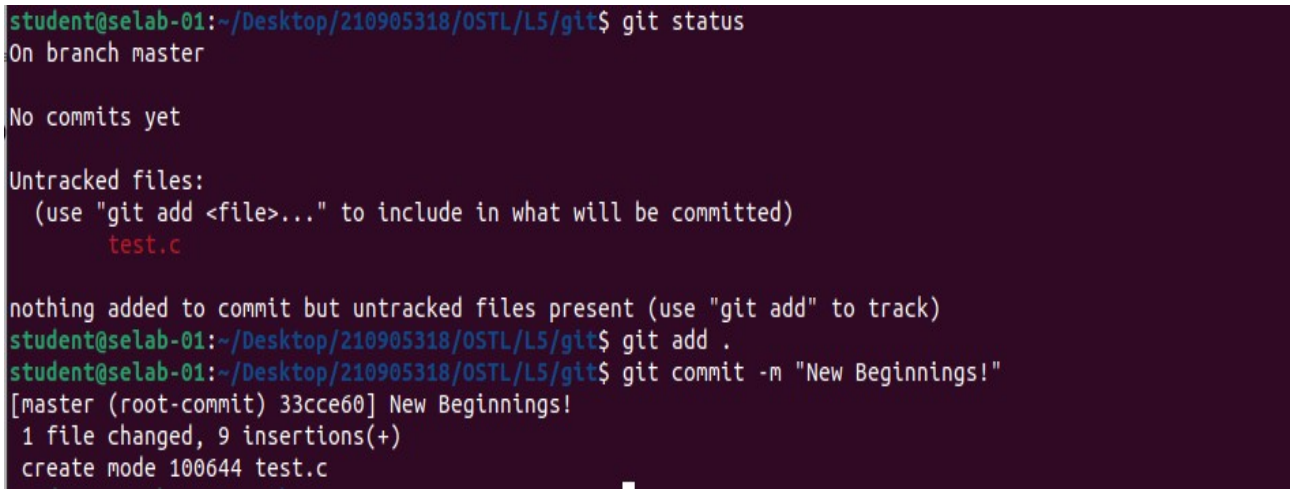
```
    return 0;
}
```

- git init (Creates empty repository)



2)Check the status of the repository. Add the file and commit the changes.

- Git status (Shows the status of the repository)
- git add . (Adds all files in current folder to commit)
- Git commit -m “” (Commits the added files)



3)Modify your C program (HelloWorld.c) such that it takes a number as input and prints a new number after adding 1 to the input. Check the changes that you have made using $ git diff

```
#include <stdio.h>

int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
        i++;
        printf("Number is %d\n", i);
        return 0;
}
```

```
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ nano test.c
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ cat test.c
#include <stdio.h>

int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
        i++;
        printf("Number is %d\n", i);
        return 0;
}
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git diff
diff --git a/test.c b/test.c
index e3add38..f2c831b 100644
--- a/test.c
+++ b/test.c
@@ -4,6 +4,7 @@ int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
+       i++;
        printf("Number is %d\n", i);
        return 0;
 }
```

4)Add this file to your git repository and commit. Check the logs of git commit

```
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git add .
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git commit -m "Second commit!"
[master 49faca5] Second commit!
 1 file changed, 1 insertion(+)
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git log
commit 49faca5a2601d6966ec16f68c1705b9e8dae714e (HEAD -> master)
Author: Johnathan Doe <johnathandoe@gmail.com>
Date:   Tue Dec 26 14:53:09 2023 +0530

    Second commit!

commit 33cce607d4641de897639623040a2d5362898141
Author: Johnathan Doe <johnathandoe@gmail.com>
Date:   Tue Dec 26 14:48:15 2023 +0530

    New Beginnings!
```

5) Unmodifying a Modified File:
a. Again, modify your C program (HelloWorld.c) such that now it adds 2 to the input before printing it. Check the git status.
b. Later you can unmodify the program by $ git checkout -- HelloWorld.c
(CAREFUL this deletes the local changes in the tracked file)
c. Check git status again.

```
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ nano test.c
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ cat test.c
#include <stdio.h>

int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
        i++;i++;
        printf("Number is %d\n", i);
        return 0;
}
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.c

no changes added to commit (use "git add" and/or "git commit -a")
```

```
        scanf("%d", &i);
        i++;i++;
        printf("Number is %d\n", i);
        return 0;
}
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.c

no changes added to commit (use "git add" and/or "git commit -a")
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git checkout -- test.c
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ git status
On branch master
nothing to commit, working tree clean
student@selab-01:~/Desktop/210905318/OSTL/L5/git$ cat test.c
#include <stdio.h>

int main(){
        int i;
        printf("Enter: ");
        scanf("%d", &i);
        i++;
        printf("Number is %d\n", i);
        return 0;
}
```

6)Checking old commit:
a. Each commit operation generates a commit ID (you can see using git log)
b. Run $ git show 7973e54895bdfb7c226952bd612fec81055305e6 (Your commit ID).