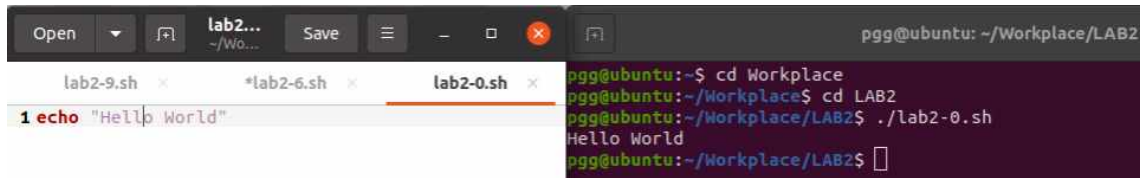


OSS - Lab2 REPORT

20213070 박건우

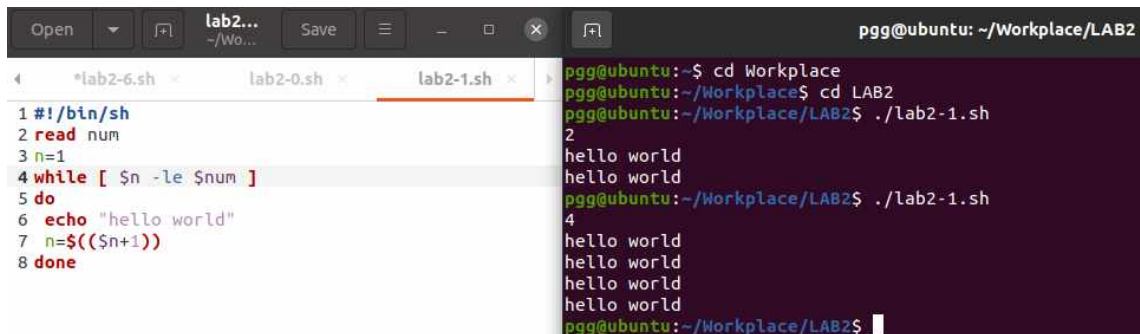
1. Lab 2-0



```
Open lab2... Save
lab2-9.sh lab2-6.sh lab2-0.sh
1 echo "Hello World"
pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-0.sh
Hello World
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-0은 매우 간단했다. 화면에 Hello world를 출력하는 것으로 echo만 사용해서 쉽게 풀 수 있었다.

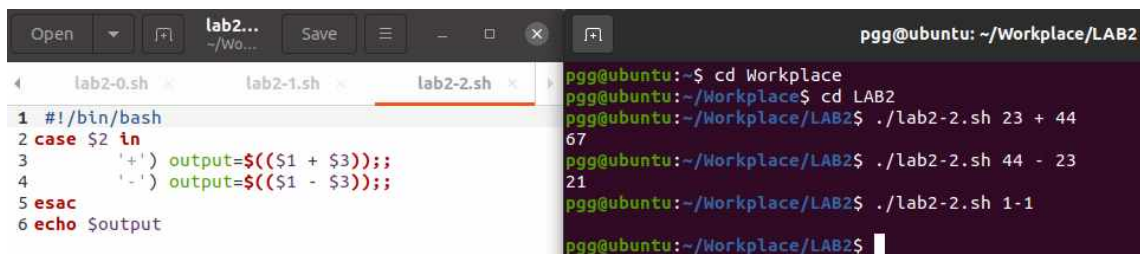
2. Lab 2-1



```
Open lab2... Save
lab2-6.sh lab2-0.sh lab2-1.sh
1 #!/bin/sh
2 read num
3 n=1
4 while [ $n -le $num ]
5 do
6 echo "hello world"
7 n=$((n+1))
8 done
pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-1.sh
2
hello world
hello world
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-1.sh
4
hello world
hello world
hello world
hello world
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-1은 사용자가 숫자를 입력해야 하므로 read를 사용한다. 입력한 숫자는 num에 저장 되고 while문을 사용해 n=1의 값이 입력한 num값보다 작거나 같을 때까지 반복되면서 hello world를 출력하게 된다.

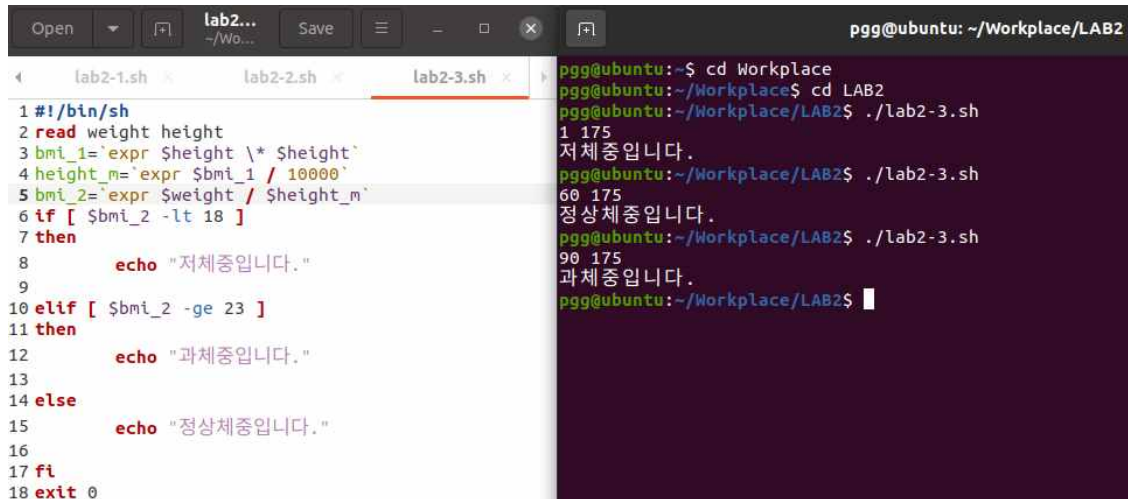
3. Lab 2-2



```
Open lab2... Save
lab2-0.sh lab2-1.sh lab2-2.sh
1 #!/bin/bash
2 case $2 in
3 '+' ) output=$(( $1 + $3 ));;
4 '-' ) output=$(( $1 - $3 ));;
5 esac
6 echo $output
pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-2.sh 23 + 44
67
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-2.sh 44 - 23
21
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-2.sh 1-1
0
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-2는 간단한 계산기를 만드는 문제였다. 입력받은 연산기호에 따라서 계산값이 달라지므로 case를 사용해서 구현했다. case를 이용해 두 번째로 입력받은 문자가 '+'인 경우에는 두 숫자를 더해주고 '-'인 경우에는 두 숫자를 빼준다. case가 끝날 때, esac를 붙이는 것에 주의했다. 덧셈과 뺄셈 과정에서는 앞으로 나올 문제에서 자주 사용하게 될 'expr'대신 다른 방식으로 계산해보았다.

4. Lab 2-3

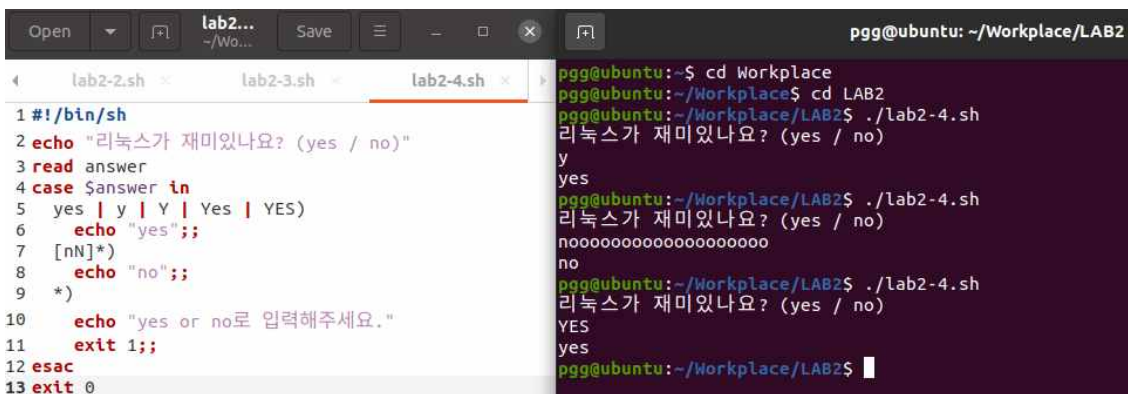


```
1 #!/bin/sh
2 read weight height
3 bmi_1=`expr $height \* $height`
4 height_m=`expr $bmi_1 / 10000`
5 bmi_2=`expr $weight / $height_m`
6 if [ $bmi_2 -lt 18 ]
7 then
8     echo "저체중입니다."
9
10 elif [ $bmi_2 -ge 23 ]
11 then
12     echo "과체중입니다."
13
14 else
15     echo "정상체중입니다."
16
17 fi
18 exit 0
```

```
pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-3.sh
1 175
저체중입니다.
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-3.sh
60 175
정상체중입니다.
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-3.sh
90 175
과체중입니다.
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-3은 입력한 몸무게와 키를 바탕으로 bmi를 계산한 뒤, 자신이 정상체중인지 검사해주는 프로그램이다. 입력받은 값을 계산해주기 위해서는 `expr`을 사용해 주었다. 그리고 정상체중 여부를 확인하는 과정은 if문으로 구현했다. if문을 활용해 정상체중 범위에 못 미치거나 벗어나는 경우는 각각 “저체중입니다”와 “과체중입니다”를 출력하도록 해주었고, 정상체중인 경우는 else를 사용해 출력해주도록 했다. if문을 사용하는 과정에서 then과 fi를 잊어버리는 것에 주의했다.

5. Lab 2-4

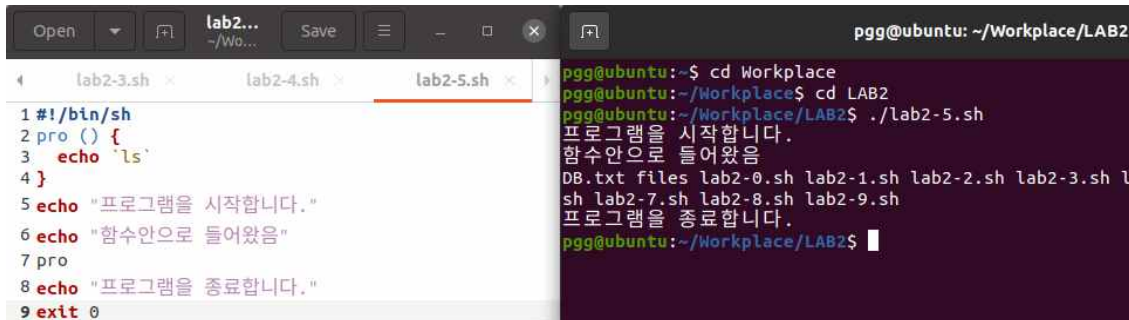


```
1 #!/bin/sh
2 echo "리눅스가 재미있나요? (yes / no)"
3 read answer
4 case $answer in
5     yes | y | Y | Yes | YES)
6         echo "yes";;
7     [nN]*)
8         echo "no";;
9     *)
10        echo "yes or no로 입력해주세요."
11        exit 1;;
12 esac
13 exit 0
```

```
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-4.sh
리눅스가 재미있나요? (yes / no)
y
yes
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-4.sh
리눅스가 재미있나요? (yes / no)
noooooooooooooooooooooo
no
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-4.sh
리눅스가 재미있나요? (yes / no)
YES
yes
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-4는 입력받은 값이 정해진 틀의 값과 살짝 달라도 자동으로 인식해서 yes인지 no인지 구별해주는 프로그램이다. 먼저 read를 통해 입력받은 answer값이 y, Y, Yes, YES여도 case에서 |를 사용해 yes를 출력하도록 해주었다. 그리고 no인 경우는 [nN]*를 사용해서 입력한 문자열이 n또는 N으로 시작하기만 한다면 no를 출력하도록 해주었다. case에서는 |를 사용해서 허용되는 값을 추가해주는 것에 주의했다.

6. Lab 2-5

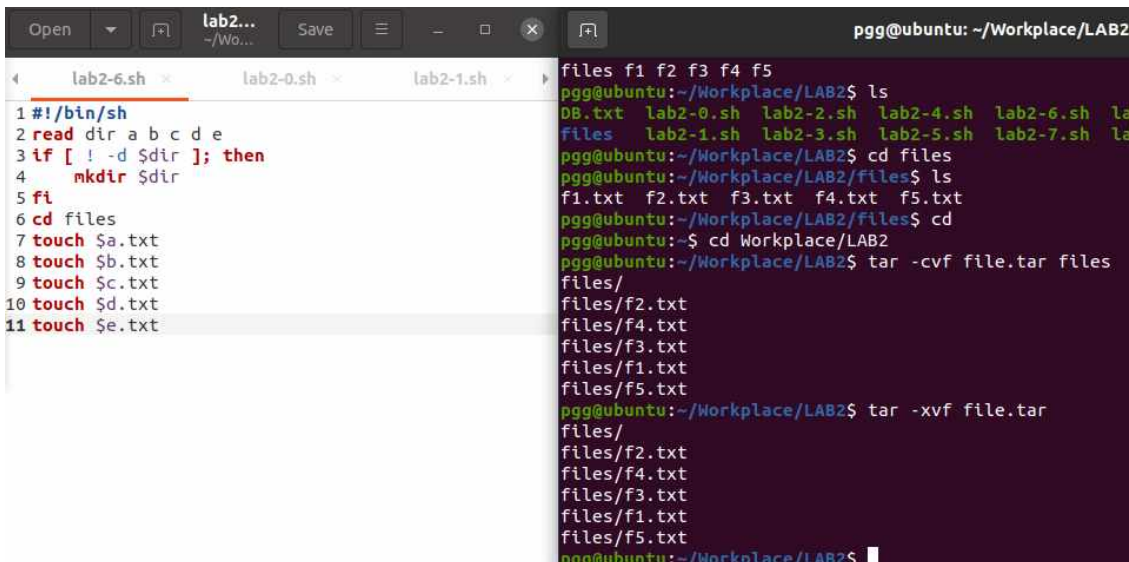


```
lab2-5.sh
1 #!/bin/sh
2 pro () {
3   echo `ls`
4 }
5 echo "프로그램을 시작합니다."
6 echo "함수안으로 들어왔습니다"
7 pro
8 echo "프로그램을 종료합니다."
9 exit 0

pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~$ cd Workplace
pgg@ubuntu:~/Workplace$ cd LAB2
pgg@ubuntu:~/Workplace/LAB2$ ./lab2-5.sh
프로그램을 시작합니다.
함수안으로 들어왔습니다
DB.txt files lab2-0.sh lab2-1.sh lab2-2.sh lab2-3.sh lab2-4.sh lab2-5.sh lab2-6.sh lab2-7.sh lab2-8.sh lab2-9.sh
프로그램을 종료합니다.
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-5는 내부함수를 만들어야 했다. 함수를 선언하기에 앞서 함수의 이름은 pro로 정해주었다. 주어진 문제에서는 리눅스 명령어인 'ls'를 실행해주는 것이기에 echo 'ls'로 함수가 실행되면 LAB2의 목록을 출력해주도록 했다. 함수를 실행하는 방법은 함수의 이름을 입력만 해주면 됐기에 큰 문제없이 문제를 풀 수 있었다.

7. Lab 2-6

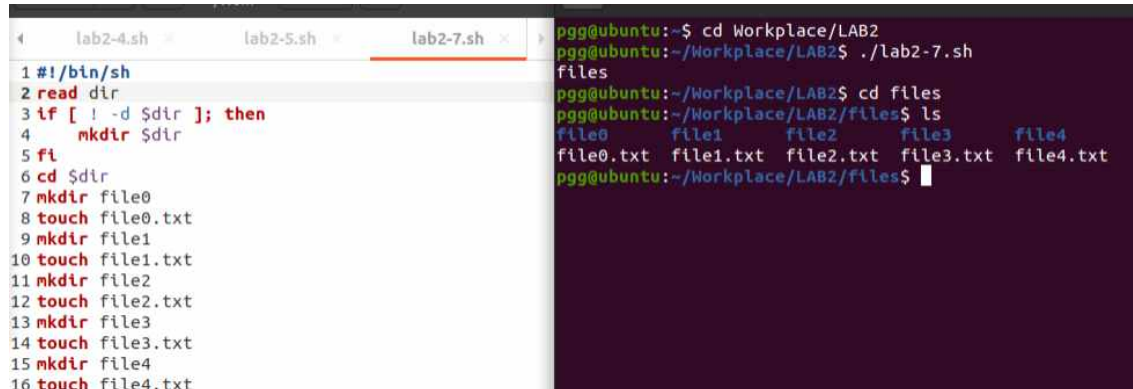


```
lab2-6.sh
1 #!/bin/sh
2 read dir a b c d e
3 if [ ! -d $dir ]; then
4   mkdir $dir
5 fi
6 cd files
7 touch $a.txt
8 touch $b.txt
9 touch $c.txt
10 touch $d.txt
11 touch $e.txt

pgg@ubuntu: ~/Workplace/LAB2
pgg@ubuntu:~/Workplace/LAB2$ ls
DB.txt lab2-0.sh lab2-2.sh lab2-4.sh lab2-6.sh lab2-8.sh
files lab2-1.sh lab2-3.sh lab2-5.sh lab2-7.sh lab2-9.sh
pgg@ubuntu:~/Workplace/LAB2$ cd files
pgg@ubuntu:~/Workplace/LAB2/files$ ls
f1.txt f2.txt f3.txt f4.txt f5.txt
pgg@ubuntu:~/Workplace/LAB2/files$ cd
pgg@ubuntu:~$ cd Workplace/LAB2
pgg@ubuntu:~/Workplace/LAB2$ tar -cvf file.tar files
files/
files/f2.txt
files/f4.txt
files/f3.txt
files/f1.txt
files/f5.txt
pgg@ubuntu:~/Workplace/LAB2$ tar -xvf file.tar
files/
files/f2.txt
files/f4.txt
files/f3.txt
files/f1.txt
files/f5.txt
pgg@ubuntu:~/Workplace/LAB2$
```

Lab 2-6은 폴더를 생성하고 그안에 txt파일을 만든 뒤 폴더를 압축하고 다시 압축 해제하는 문제이다. 먼저 만드려는 폴더가 LAB2폴더에 있는지 확인하고 없다면 폴더를 생성해주어야 하기 때문에 if문을 사용했다. 폴더를 생성할 때는 mkdir 명령어를 사용했고 read를 통해 입력받은 이름대로 폴더가 생성되었다. 사진처럼 ls를 입력했을 경우 폴더가 하나 더 생긴 것을 알 수 있다. 그리고 read를 통해 files폴더 내에 만들어야 할 txt파일들의 이름을 입력받았고 이 과정에서 txt파일이 LAB2가 아닌 files폴더에 생성되기 위해 cd files를 touch 이전에 입력해놓는 것에 주의했다. 따라서 cd files를 입력한 뒤 ls를 입력한 경우, 내가 생성한 txt파일들이 보이는 것을 알 수 있다. 그리고 폴더를 압축하기 위해서 -cvf를 사용했고 이를 다시 압축 해제하기 위해서는 -xvf를 사용했다.

8. Lab 2-7



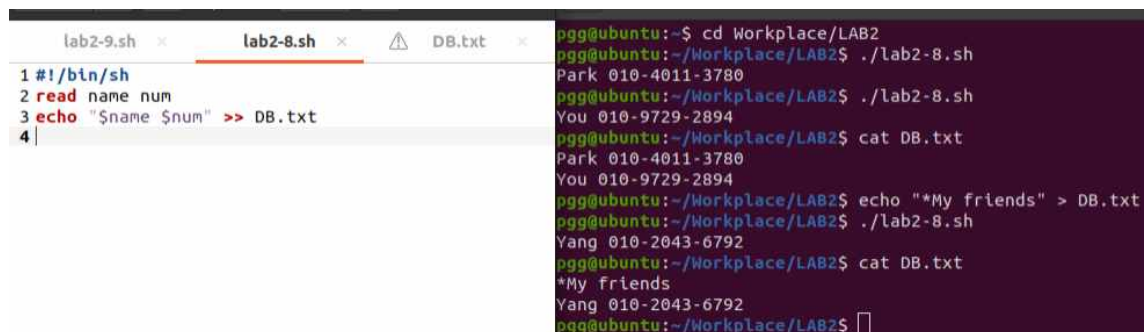
The screenshot shows a script editor on the left with a file named 'lab2-7.sh'. The script contains the following code:

```
1 #!/bin/sh
2 read dir
3 if [ ! -d $dir ]; then
4     mkdir $dir
5 fi
6 cd $dir
7 mkdir file0
8 touch file0.txt
9 mkdir file1
10 touch file1.txt
11 mkdir file2
12 touch file2.txt
13 mkdir file3
14 touch file3.txt
15 mkdir file4
16 touch file4.txt
```

On the right, a terminal window shows the execution of the script. The user runs 'cd Workplace/LAB2' and './lab2-7.sh'. The script creates a directory 'files' and then runs 'cd files'. The user then runs 'ls', which lists the files: 'file0', 'file1', 'file2', 'file3', and 'file4'. Each file has a corresponding '.txt' file created in the same directory.

먼저 폴더를 만들기 위해서는 Lab 2-6에서 했던 것과 같이 if문을 사용해 구현했다. 그리고 cd \$dir을 입력해 자동으로 그 폴더에서 사진과 같이 각각의 하위폴더와 txt파일을 생성하도록 해주었다.

9. Lab 2-8



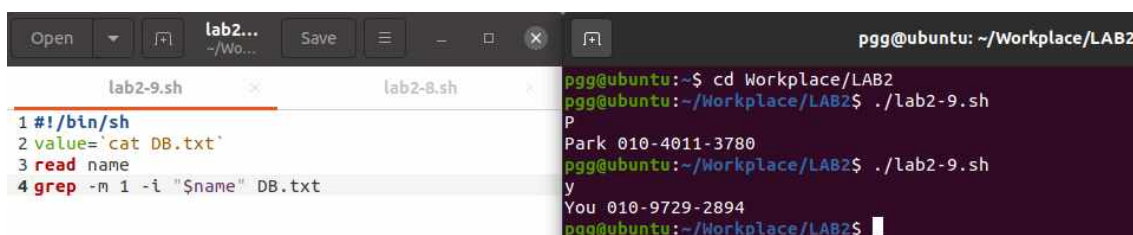
The screenshot shows a script editor on the left with a file named 'lab2-8.sh'. The script contains the following code:

```
1 #!/bin/sh
2 read name num
3 echo "$name $num" >> DB.txt
4
```

On the right, a terminal window shows the execution of the script. The user runs 'cd Workplace/LAB2' and './lab2-8.sh'. The script prompts for input, and the user enters 'Park 010-4011-3780'. The script then runs 'cat DB.txt', which shows the output: 'Park 010-4011-3780'. The user then enters 'You 010-9729-2894' and runs './lab2-8.sh' again. The script then runs 'cat DB.txt', which shows the output: 'Park 010-4011-3780' and 'You 010-9729-2894'. The user then enters '*My friends' and runs './lab2-8.sh' again. The script then runs 'cat DB.txt', which shows the output: 'Park 010-4011-3780', 'You 010-9729-2894', and '*My friends'.

Lab 2-8은 입력한 값을 생성해놓은 DB.txt파일에 기록하는 문제이다. 먼저 이름과 전화번호를 각각 읽기위해 read 명령어를 사용해주었고, 이는 >>에 의해 DB.txt에 저장된다. 그리고 "echo " " > DB.txt"를 사용해 기존 DB.txt파일에 기록된 정보들이 위의 사진에서 보이는 것과 같이 모두 지워지고 다시 추가해서 작성할 수 있다.

10. Lab 2-9



The screenshot shows a script editor on the left with a file named 'lab2-9.sh'. The script contains the following code:

```
1 #!/bin/sh
2 value='cat DB.txt'
3 read name
4 grep -m 1 -i "$name" DB.txt
```

On the right, a terminal window shows the execution of the script. The user runs 'cd Workplace/LAB2' and './lab2-9.sh'. The script prompts for input, and the user enters 'P'. The script then runs 'cat DB.txt', which shows the output: 'Park 010-4011-3780'. The user then enters 'y' and runs './lab2-9.sh' again. The script then runs 'cat DB.txt', which shows the output: 'Park 010-4011-3780' and 'You 010-9729-2894'.

Lab 2-9는 이름을 입력했을 때, 해당 사람의 이름과 전화번호를 출력해주는 프로그램이다. 먼저 DB.txt의 값들을 읽기 위해 value에 저장 해주었고, 읽은 이름에 따라 DB.txt에서 해당 행을 출력해줄도록 했다. 이때 입력한 문자가 한 사람의 이름과만 일치해도 모든 사람의 정보가 출력되는 문제를 해결하기 위해 Line4에서 "-m 1"을 추가해 해당되는 한 행만 출력하도록 해주었다.