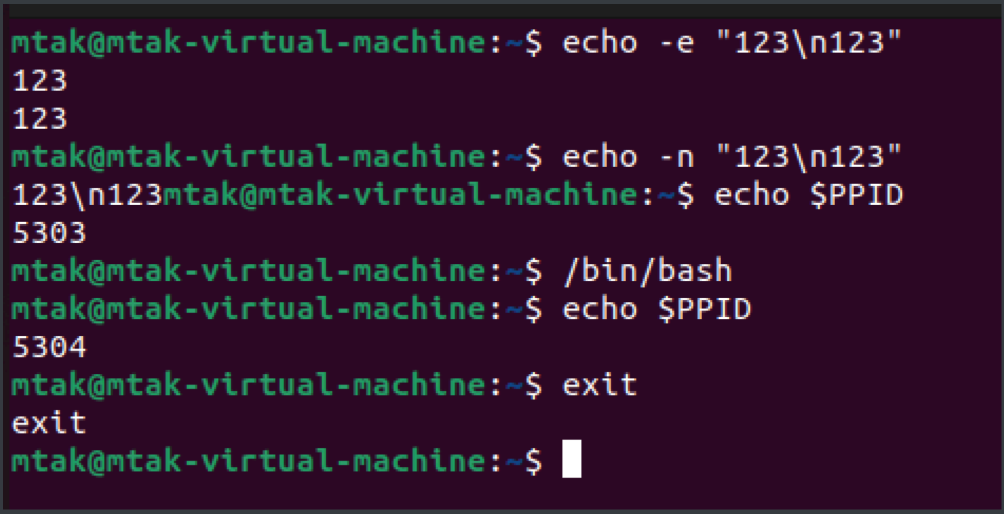
Homework #8

1. Do “clear”, then Do “echo –e ‘123\n123’”, then do “echo –n ‘123\n123’, then do “echo $PPID”, then do “/bin/bash”, then do “echo $PPID”, then do “exit”

- Take a screen shot

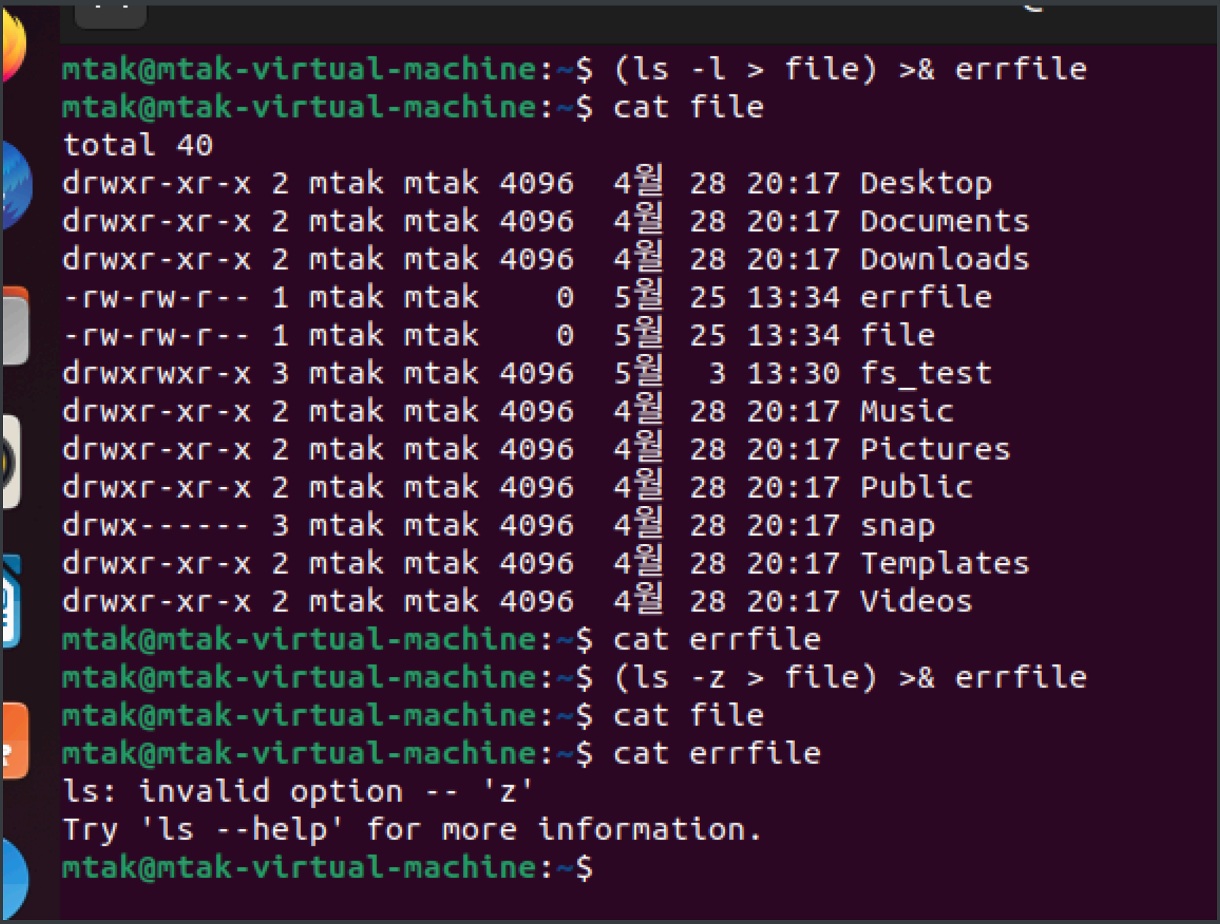


```
mtak@mtak-virtual-machine:~$ echo -e "123\n123"
123
123
mtak@mtak-virtual-machine:~$ echo -n "123\n123"
123\n123mtak@mtak-virtual-machine:~$ echo $PPID
5303
mtak@mtak-virtual-machine:~$ /bin/bash
mtak@mtak-virtual-machine:~$ echo $PPID
5304
mtak@mtak-virtual-machine:~$ exit
exit
mtak@mtak-virtual-machine:~$ 
```

- Why the result of “echo $PPID” is different? Why does your terminal still work after doing “exit”?
  /bin/bash를 실행하면 자식 프로세스가 새로 생기기 때문에 부모 pid가 바뀐다. exit명령어는 방금 실행한 자식 프로세스를 종료하기 때문에 부모 프로세스인 bash로 실행 권한이 넘어오게 된다.

2. Do “clear”, then do “(ls –l > file) >& errfile”, then do “cat file”, then do “cat errfile”, then do “(ls – z > file) >& errfile”, then do “cat file”, then do “cat errfile”.

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ (ls -l > file) >& errfile
mtak@mtak-virtual-machine:~$ cat file
total 40
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Desktop
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Documents
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Downloads
-rw-rw-r-- 1 mtak mtak    0  5월 25 13:34 errfile
-rw-rw-r-- 1 mtak mtak    0  5월 25 13:34 file
drwxrwxr-x 3 mtak mtak 4096  5월  3 13:30 fs_test
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Music
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Pictures
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Public
drwx------ 3 mtak mtak 4096  4월 28 20:17 snap
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Templates
drwxr-xr-x 2 mtak mtak 4096  4월 28 20:17 Videos
mtak@mtak-virtual-machine:~$ cat errfile
mtak@mtak-virtual-machine:~$ (ls -z > file) >& errfile
mtak@mtak-virtual-machine:~$ cat file
mtak@mtak-virtual-machine:~$ cat errfile
ls: invalid option -- 'z'
Try 'ls --help' for more information.
mtak@mtak-virtual-machine:~$
```

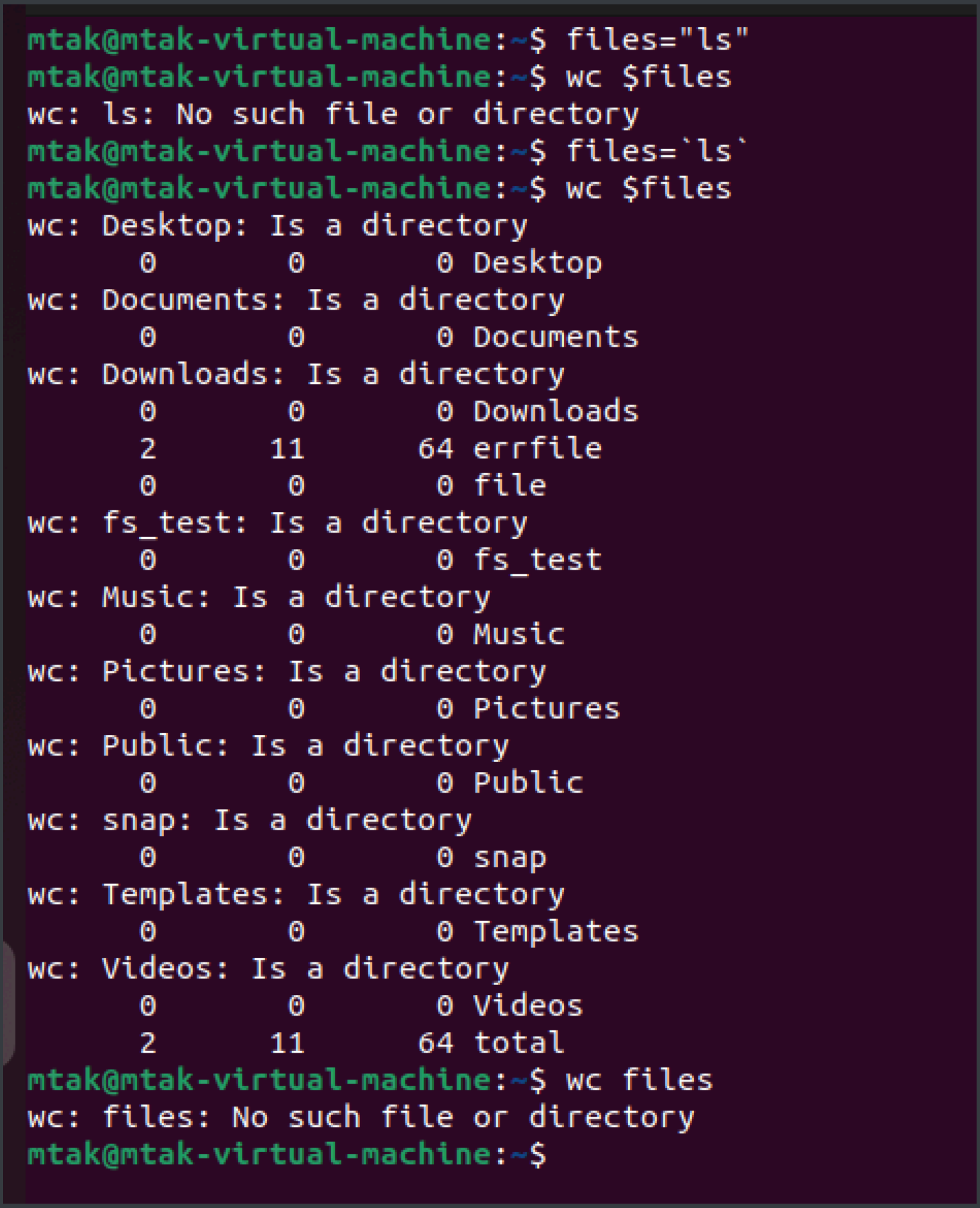
- What is the difference? Why?

  첫 번째 명령어 ls -l 은 정상적으로 실행이 되므로, 출력물인 현재 디렉토리의 list가 file로 redirection된다. error가 없었기 때문에 errfile로 리다이랙션 되는 출력물은 없다.

  하지만 두 번째 명령어 ls -z 에서 -z는 정의되지 않은 옵션이므로, 에러를 일으킨다. 따라서 정상적인 출력물은 없어서 빈 문자열이 file에 overwrite되고, 에러문은 errfile에 overwrite된다.

3. Do “clear”, then do “files=’ls’” (with quotes), then do “wc $files”, then do “files= `ls` ” (back quotes), then do “wc files”, then do “wc $files”.

- Take a screen shot



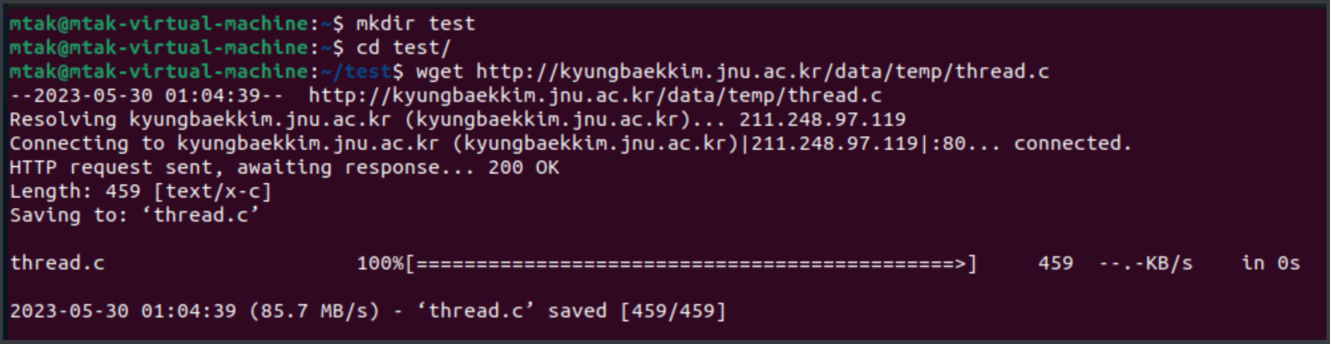
```
mtak@mtak-virtual-machine:~$ files="ls"
mtak@mtak-virtual-machine:~$ wc $files
wc: ls: No such file or directory
mtak@mtak-virtual-machine:~$ files=`ls`
mtak@mtak-virtual-machine:~$ wc $files
wc: Desktop: Is a directory
      0       0       0 Desktop
wc: Documents: Is a directory
      0       0       0 Documents
wc: Downloads: Is a directory
      0       0       0 Downloads
      2      11      64 errfile
      0       0       0 file
wc: fs_test: Is a directory
      0       0       0 fs_test
wc: Music: Is a directory
      0       0       0 Music
wc: Pictures: Is a directory
      0       0       0 Pictures
wc: Public: Is a directory
      0       0       0 Public
wc: snap: Is a directory
      0       0       0 snap
wc: Templates: Is a directory
      0       0       0 Templates
wc: Videos: Is a directory
      0       0       0 Videos
      2      11      64 total
mtak@mtak-virtual-machine:~$ wc files
wc: files: No such file or directory
mtak@mtak-virtual-machine:~$
```

- Explain the result
  처음 single quote 로 감싼 건 String이다. 따라서 wc $files는 wc ls 로 해석되어 에러를 유발한다. 두번째는 back quote로 감싼 ls는 명령어를 의미한다. wc files는 files를 변수로 생각하지 않고 files리터럴 자체를 argument 로 받으므로 오류가 난다.

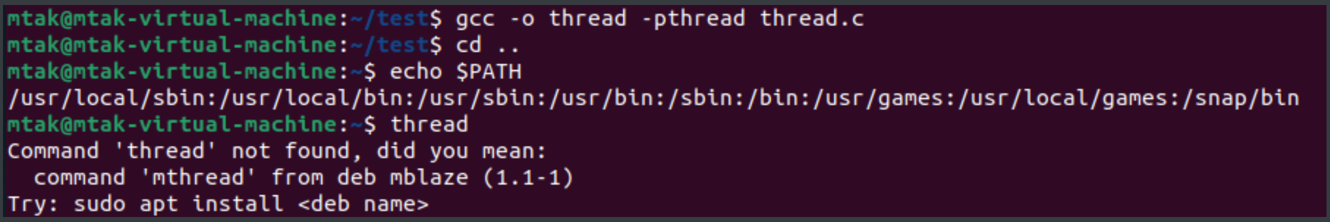
  wc $files는 해당 ls 명령어가 정상적으로 wc에 전달되어 결과값이 출력됨을 볼 수 있다

4. Do “clear”, then do “mkdir test”, then do “cd test”, then do “wget http://kyungbaekkim.jnu.ac.kr/data/temp/thread.c”, then do “gcc –o thread –pthread thread.c”, then do “cd ..”, then do “echo $PATH”, then do “thread”, then do “PATH=”./test:$PATH”, then do “echo $PATH”, then do “thread”

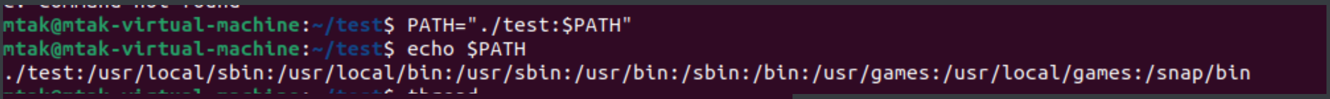
- Take a screen shot



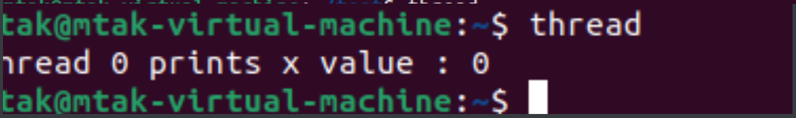
```
mtak@mtak-virtual-machine:~$ mkdir test
mtak@mtak-virtual-machine:~$ cd test/
mtak@mtak-virtual-machine:~/test$ wget http://kyungbaekkim.jnu.ac.kr/data/temp/thread.c
--2023-05-30 01:04:39--  http://kyungbaekkim.jnu.ac.kr/data/temp/thread.c
Resolving kyungbaekkim.jnu.ac.kr (kyungbaekkim.jnu.ac.kr)... 211.248.97.119
Connecting to kyungbaekkim.jnu.ac.kr (kyungbaekkim.jnu.ac.kr)|211.248.97.119|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 459 [text/x-c]
Saving to: ‘thread.c’

thread.c                    100%[=============================================>]     459  --.-KB/s    in 0s

2023-05-30 01:04:39 (85.7 MB/s) - ‘thread.c’ saved [459/459]
```



```
mtak@mtak-virtual-machine:~/test$ gcc -o thread -pthread thread.c
mtak@mtak-virtual-machine:~/test$ cd ..
mtak@mtak-virtual-machine:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
mtak@mtak-virtual-machine:~$ thread
Command 'thread' not found, did you mean:
  command 'mthread' from deb mblaze (1.1-1)
Try: sudo apt install <deb name>
```



```
mtak@mtak-virtual-machine:~/test$ PATH="./test:$PATH"
mtak@mtak-virtual-machine:~/test$ echo $PATH
./test:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```



```
tak@mtak-virtual-machine:~$ thread
hread 0 prints x value : 0
tak@mtak-virtual-machine:~$ █
```

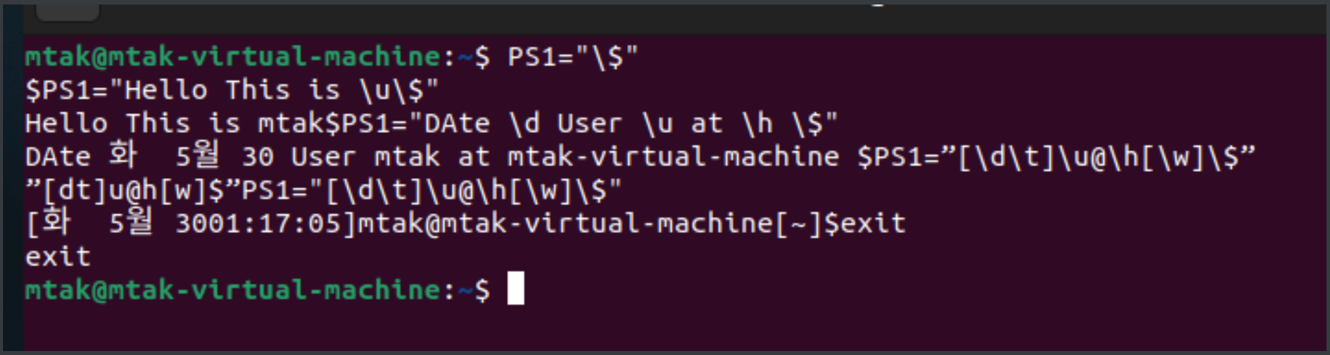
- Why the results of doing “thread” are different? Explain it.

  처음은 PATH에 thread가 있는 경로를 설정해주지않았다.

  하지만 두번째 thread는 thread파일이 있는 경로를 PATH에 추가했으므로(./test) 정상적으로 실행됨을 알 수 있다.

5. Do “bash”, then do “clear”, then do “PS1=”$””, then do “PS1=”Hello This is \u$””, then do “PS1=”Date \d User \u at \h $””, then do “PS1=”[\d\t]\u@\h[\w]$””, then do “exit”

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ PS1="\$"
$PS1="Hello This is \u\$"
Hello This is mtak$PS1="DAte \d User \u at \h \$"
DAte 화  5월 30 User mtak at mtak-virtual-machine $PS1=”[\d\t]\u@\h[\w]\$”
”[dt]u@h[w]$”PS1="[\d\t]\u@\h[\w]\$"
[화  5월 3001:17:05]mtak@mtak-virtual-machine[~]$exit
exit
mtak@mtak-virtual-machine:~$ █
```

- Explain why the prompt changes in different forms.

  PS1 변수는 Prompt Style을 지정하는 변수이다.

  첫 번째 명령어에서 \$는 현재 uid가 root가 아닌 이상 $를 의미한다.

두 번째 명령어에서는 \u 는 User name인, 여기서는 mtak을 의미하고, $는 루트 유저는 #, 다른 유저는 $를 쉘에 나타낸다.

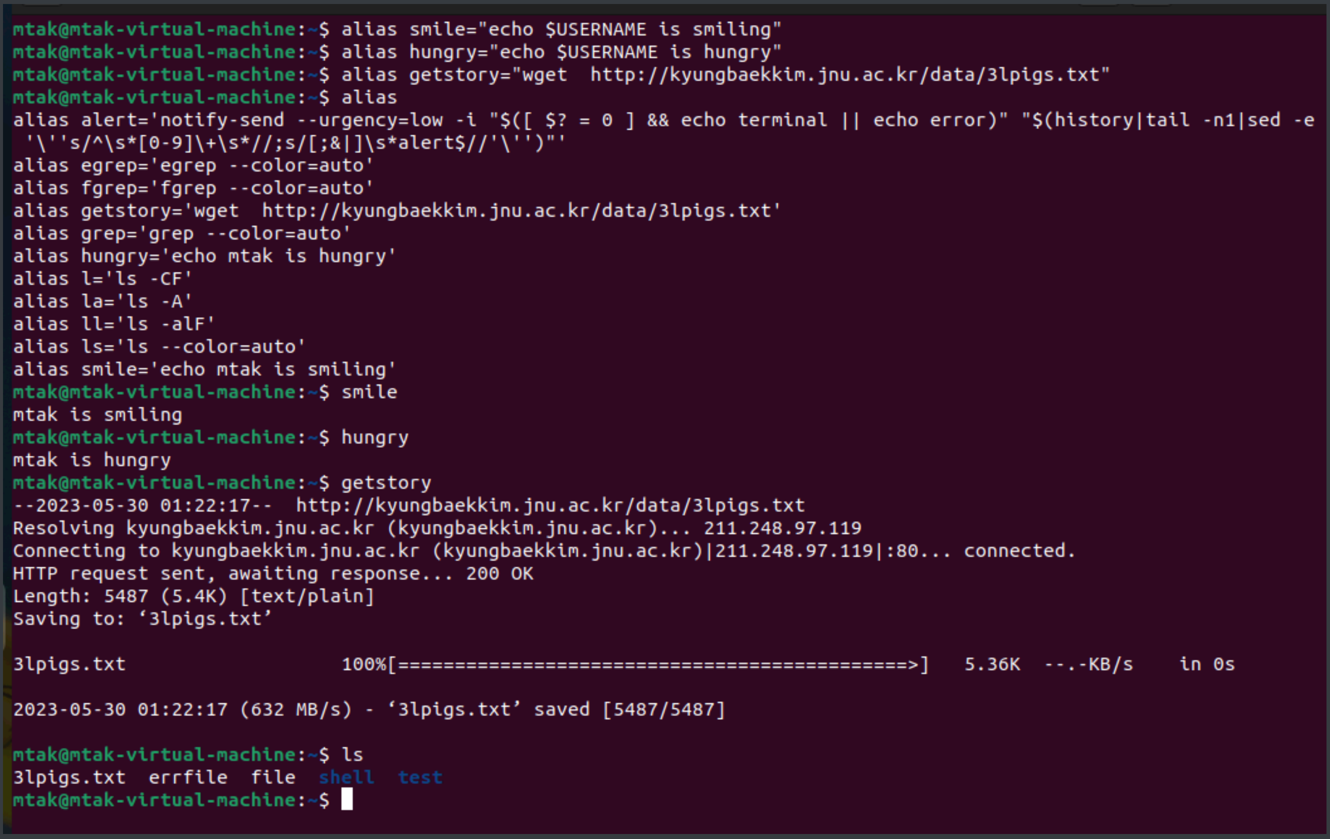
세 번째 명령어에서 \d는 날짜(주,월,일)를 나타낸다. \h는 host name인 여기서는 mtak-virtual-machine을 나타낸다.

마지막 명령어에서는 \t는 24시간으로 시간을 나타내고(시시:분분:초초) , \w는 현재 디렉토리의 절대 경로를 나타낸다.

나머지 String들은 전부 리터럴 그대로 출력된다.

6. Do “clear”, then do “alias smile=”echo $USERNAME is smiling””, then do “alias hungry=”echo $USERNAME is hungry”, then do “alias getstory=”wget http://kyungbaekkim.jnu.ac.kr/data/3lpigs.txt”, then do “alias”, then do “smile”, then do “hungry”, then do “getstory”, then do “ls”.

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ alias smile="echo $USERNAME is smiling"
mtak@mtak-virtual-machine:~$ alias hungry="echo $USERNAME is hungry"
mtak@mtak-virtual-machine:~$ alias getstory="wget  http://kyungbaekkim.jnu.ac.kr/data/3lpigs.txt"
mtak@mtak-virtual-machine:~$ alias
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e
 '\''s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias getstory='wget  http://kyungbaekkim.jnu.ac.kr/data/3lpigs.txt'
alias grep='grep --color=auto'
alias hungry='echo mtak is hungry'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
alias smile='echo mtak is smiling'
mtak@mtak-virtual-machine:~$ smile
mtak is smiling
mtak@mtak-virtual-machine:~$ hungry
mtak is hungry
mtak@mtak-virtual-machine:~$ getstory
--2023-05-30 01:22:17--  http://kyungbaekkim.jnu.ac.kr/data/3lpigs.txt
Resolving kyungbaekkim.jnu.ac.kr (kyungbaekkim.jnu.ac.kr)... 211.248.97.119
Connecting to kyungbaekkim.jnu.ac.kr (kyungbaekkim.jnu.ac.kr)|211.248.97.119|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5487 (5.4K) [text/plain]
Saving to: ‘3lpigs.txt’

3lpigs.txt                 100%[=============================================>]   5.36K  --.-KB/s    in 0s

2023-05-30 01:22:17 (632 MB/s) - ‘3lpigs.txt’ saved [5487/5487]

mtak@mtak-virtual-machine:~$ ls
3lpigs.txt  errfile  file  shell  test
mtak@mtak-virtual-machine:~$ 
```

- What are the result of doing “smile”, “hungry”, and “getstory”? why?

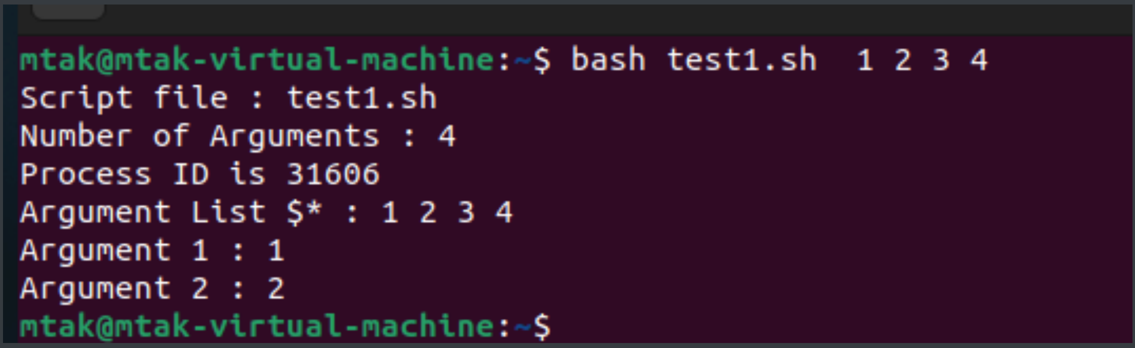
결과는 위 스크린샷과 같다. 추가 하자면 getstory에서 3lpigs.txt를 다운로드 받았기에 ls로 확인할 수 있다.

7. Create a file “test1.sh” containing following shell programming codes.

```bash
#reading arguments
#!/bin/bash
echo "Script file : $0"
echo "Number of Arguments : $#"
echo "Process ID is $$"
echo "Argument List \$* : $*"
echo "Argument 1 : $1"
echo "Argument 2 : $2"
```

Then do “clear”, then do “bash test1.sh 1 2 3 4”. *

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ bash test1.sh  1 2 3 4
Script file : test1.sh
Number of Arguments : 4
Process ID is 31606
Argument List $* : 1 2 3 4
Argument 1 : 1
Argument 2 : 2
mtak@mtak-virtual-machine:~$
```

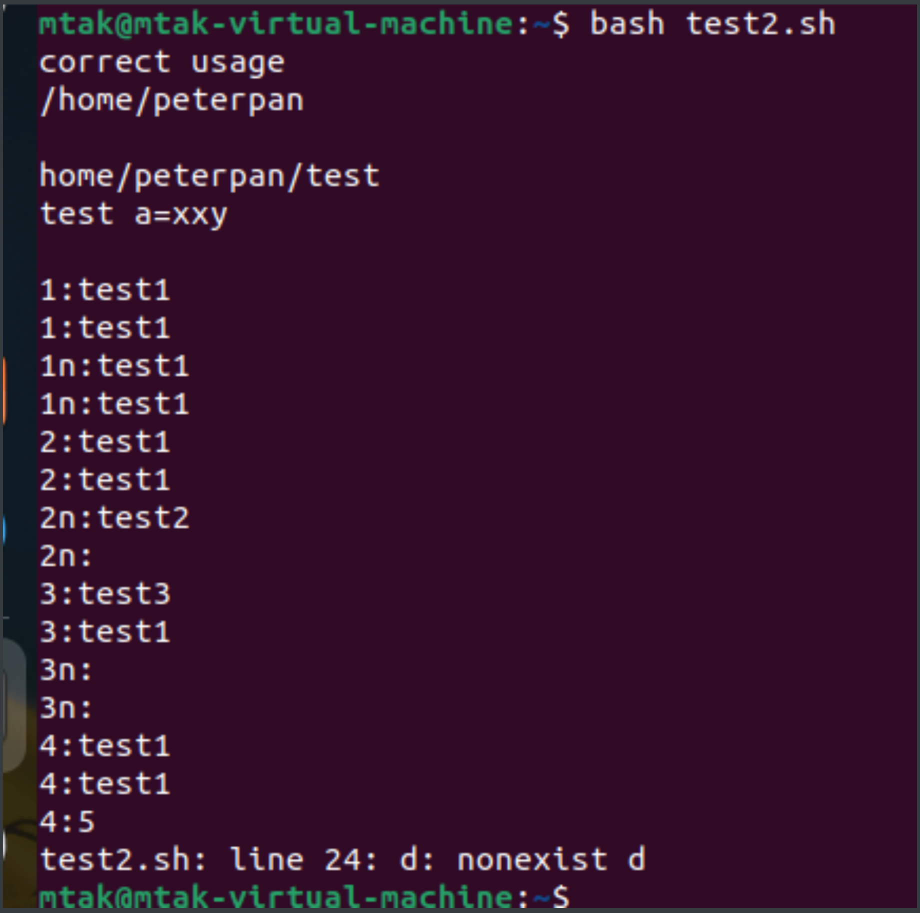
- What happens if you add a line “echo $3”?
  3이 출력된다. 왜나하면 argument 3번째 인자로 3이 전달되었기 때문이다.

8. Create a file “test2.sh” containing following shell programming codes.

```bash
#!/bin/bash
k=/home/peterpan/test
echo "correct usage"
echo ${k%/*}
echo ${k%%/*}
echo ${k#*/}
echo ${k##*/} a="xxy"
echo "$a"
echo "1:${a:="test1"}"
echo "1:$a"
echo "1n:${x:="test1"}"
echo "1n:$x"
echo "2:${a:-"test2"}"
echo "2:$a"
echo "2n:${b:-"test2"}"
echo "2n:$b"
echo "3:${a:+"test3"}"
echo "3:$a"
echo "3n:${c:+"test3"}"
echo "3n:$c"
echo "4:${a:?"test4"}"
echo "4:$a"
echo "4:${#a}"
echo "4n:${d:?"nonexist d"}"
```

Then do “clear”, then do “bash test2.sh”.

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ bash test2.sh
correct usage
/home/peterpan

home/peterpan/test
test a=xxy

1:test1
1:test1
1n:test1
1n:test1
2:test1
2:test1
2n:test2
2n:
3:test3
3:test1
3n:
3n:
4:test1
4:test1
4:5
test2.sh: line 24: d: nonexist d
mtak@mtak-virtual-machine:~$
```

9. Create a file “for.sh” containing following shell programming codes.

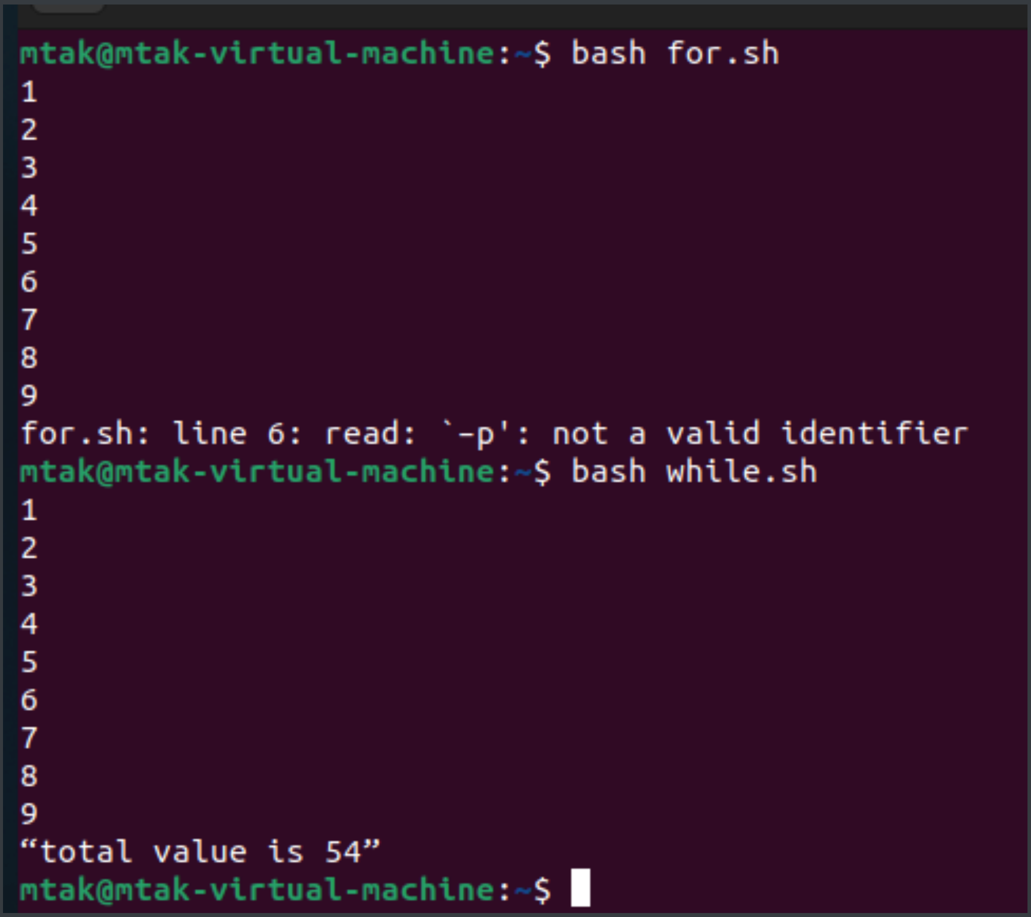
```
#!/bin/bash
for var in 1 2 3 4 5 6 7 8 9
do
echo $var
done
read -p "number :" x echo $x
```

Then, create a file “while.sh” containing following shell programming codes.

```
#!/bin/bash
var=1
tvar=0
while [ "$var" -le 9 ]
do
echo $var
var=$(($var+1))
tvar=$(($tvar+$var))
done
echo “total value is $tvar”
```

Then do “clear”, then do “bash for.sh”, then do “bash while.sh”.

- Take a screen shot



```
mtak@mtak-virtual-machine:~$ bash for.sh
1
2
3
4
5
6
7
8
9
for.sh: line 6: read: `-p': not a valid identifier
mtak@mtak-virtual-machine:~$ bash while.sh
1
2
3
4
5
6
7
8
9
"total value is 54"
mtak@mtak-virtual-machine:~$ 
```

- What happens if you change the while condition to [“$var” –lt 9]?

  le는 less equal 이므로, 9보다 작거나 같은 값까지 참이다.

  하지만 lt 는 less than이라는 뜻으로, 9보다 작은 값만 허용한다.

  따라서 1부터 8까지 출력하고, total value인 2부터 9까지합인 44 를 출력한다.

10. Create a file “main.sh” containing following shell programming codes.

```
#!/bin/bash
name=peterpan
location=neverland
print_name()
{
  echo "name: $name"
}
print_all()
{
```

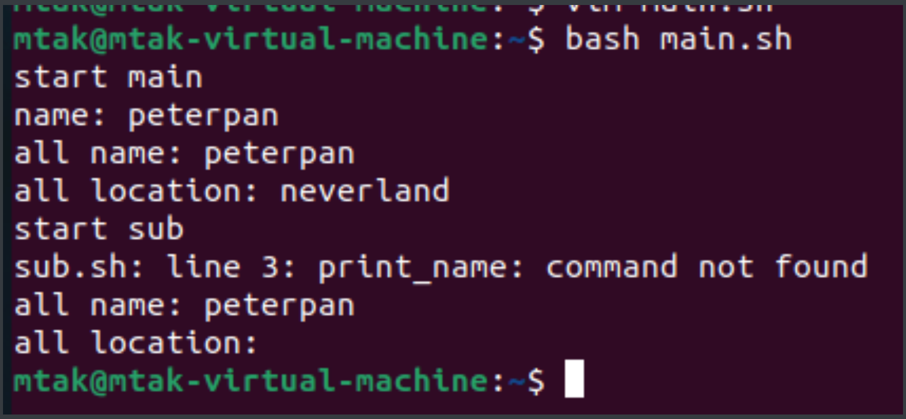
```
echo "all name: $name"
echo "all location: $location"
}
echo "start main"
print_name
print_all
export name
export -f print_all
bash sub.sh

```

Then, create a file “sub.sh” containing following shell programming codes.

```
#!/bin/bash
echo "start sub"
print_name
print_all
```

Then do “clear”, then do “bash main.sh”

- Take a screenshot



```
mtak@mtak-virtual-machine:~$ bash main.sh
start main
name: peterpan
all name: peterpan
all location: neverland
start sub
sub.sh: line 3: print_name: command not found
all name: peterpan
all location:
mtak@mtak-virtual-machine:~$
```

- Explain the results.
  main에서 export한 것은 name과 print_all 뿐이여서, sub.sh 를 bash에서 실행할때 print_name 과 location을 못 찾았다

- How to correctly use print_name on sub.sh?
  main에서 export print_name을 추가해주면 된다 `export -f print_name`

Problems

1. Bash의 프롬프트를 다음과 같이 보이게 하는 명령어는?

```
(07:09:07)Hello kbkim@ubuntu[~]$cd vi_homework/
(07:09:10)Hello
kbkim@ubuntu[~/vi_homework]$su
Password:
root@ubuntu:/home/kbkim/vi_homework#
```

 문제에서 보면 현재 유저인 kbkim의 Prompt Style은 바뀌었지만, root 유저는 안바뀌었음을 알 수 있다.
다음과 같은 명령어를 실행한다.

`PS1="(\t)Hello \u@\h[\w]\$"`

\t 는 현재 시간을 시시:분분:초초 로 나타내고,

\u 는 Username을 나타낸다.

\h 는 hostname(컴퓨터 이름)을 나타내고

\w는 현재 디렉토리의 절대경로를 나타내고,

$ 는 일반 유저면 $ 를 출력한다.(root는 #)

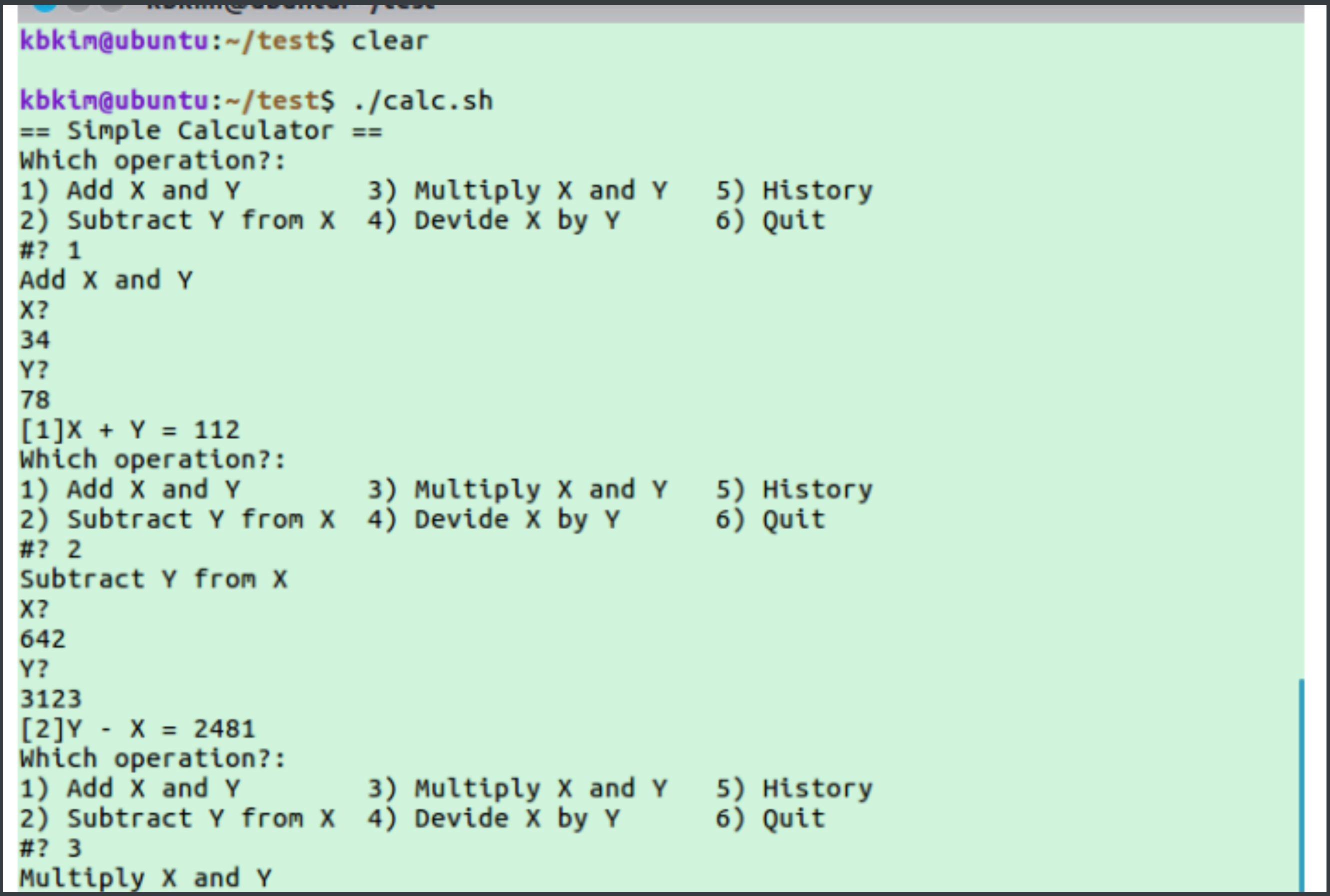
2. 다음은 .profile이라는 configuration file의 일부분이다. 이 configuration 파일이 수행하는 내용 을 설명하시오.

```
# if running bash
if [ -n "$BASH_VERSION" ];
then
# include .bashrc if it exists
if [ -f "$HOME/.bashrc" ]; then
."$HOME/.bashrc"
fi
fi
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
PATH="$HOME/bin:$PATH"
fi
```

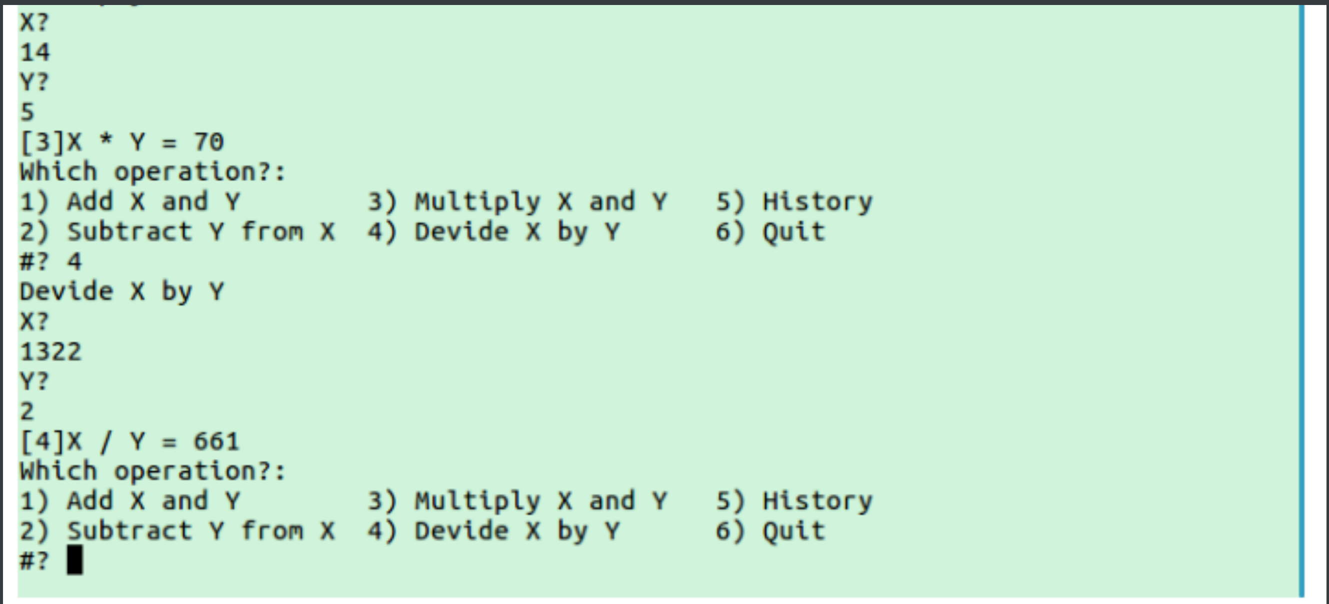
첫번째로 BASH_VERSION 이 null 이 아닌지 판단한다. 그러고 null 이 아니면, $HOME/.bashrc 의 파일이 존재하고 regular file인지 확인한다. 만약 맞다면, . $HOME/.bashrc 를 호출한다.

그 다음 if [-d "$HOME/bin"] ; 은 유저의 HOME에 private한 bin 디렉토리가 존재하는지 체크한다. 만약존재한다면, PATH에 private bin 디렉토리를 추가한다. (PATH="$HOME/bin:$PATH")

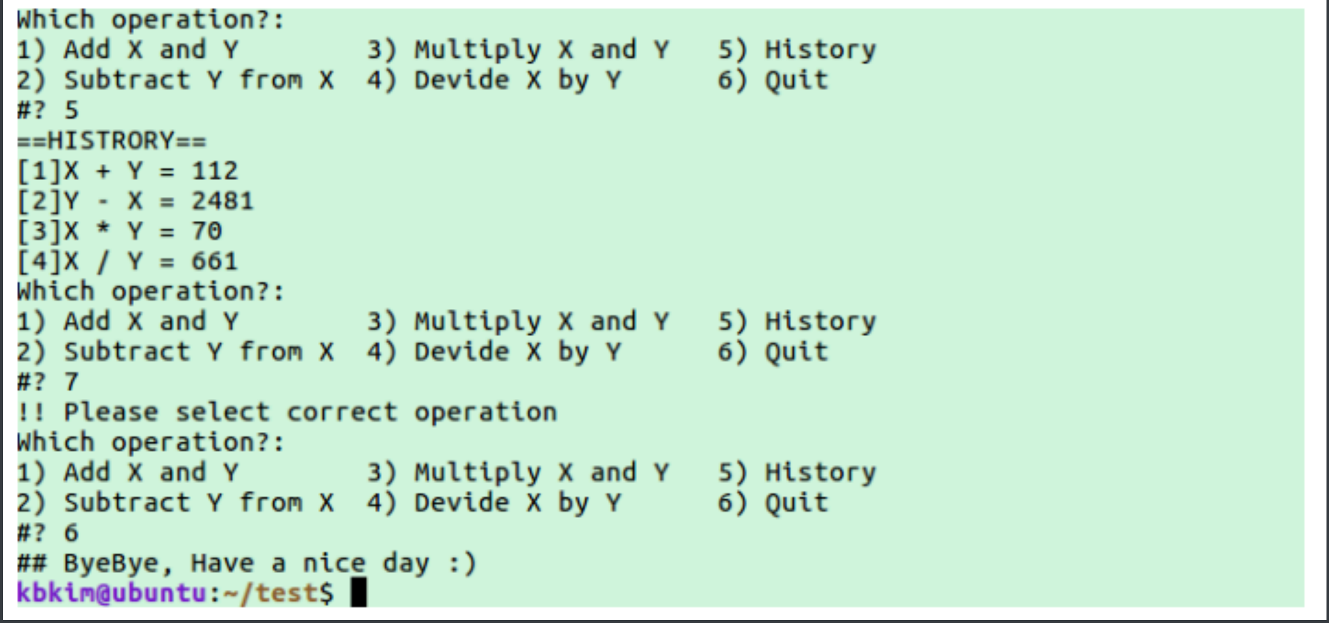
3. 다음의 화면과 같이 동작하는 calc.sh를 완성하고 동작결과를 스크린샷으로 제출하시오.



```
kbkim@ubuntu:~/test$ clear

kbkim@ubuntu:~/test$ ./calc.sh
== Simple Calculator ==
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X   4) Devide X by Y      6) Quit
#? 1
Add X and Y
X?
34
Y?
78
[1]X + Y = 112
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X   4) Devide X by Y      6) Quit
#? 2
Subtract Y from X
X?
642
Y?
3123
[2]Y - X = 2481
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X   4) Devide X by Y      6) Quit
#? 3
Multiply X and Y
```



```
X?
14
Y?
5
[3]X * Y = 70
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X  4) Devide X by Y       6) Quit
#? 4
Devide X by Y
X?
1322
Y?
2
[4]X / Y = 661
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X  4) Devide X by Y       6) Quit
#? 
```

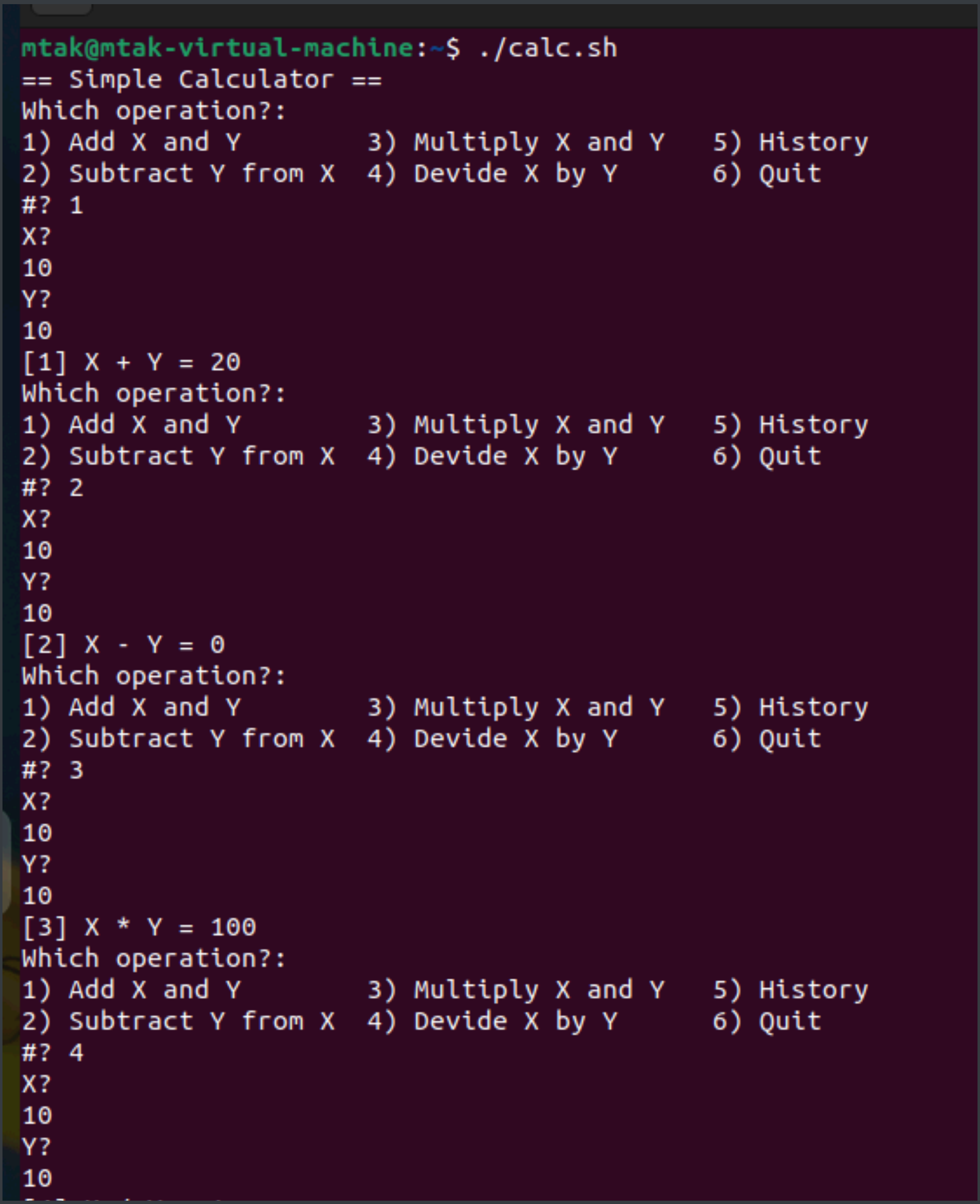


```
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X  4) Devide X by Y       6) Quit
#? 5
==HISTRORY==
[1]X + Y = 112
[2]Y - X = 2481
[3]X * Y = 70
[4]X / Y = 661
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X  4) Devide X by Y       6) Quit
#? 7
!! Please select correct operation
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X  4) Devide X by Y       6) Quit
#? 6
## ByeBye, Have a nice day :)
kbkim@ubuntu:~/test$ 
```

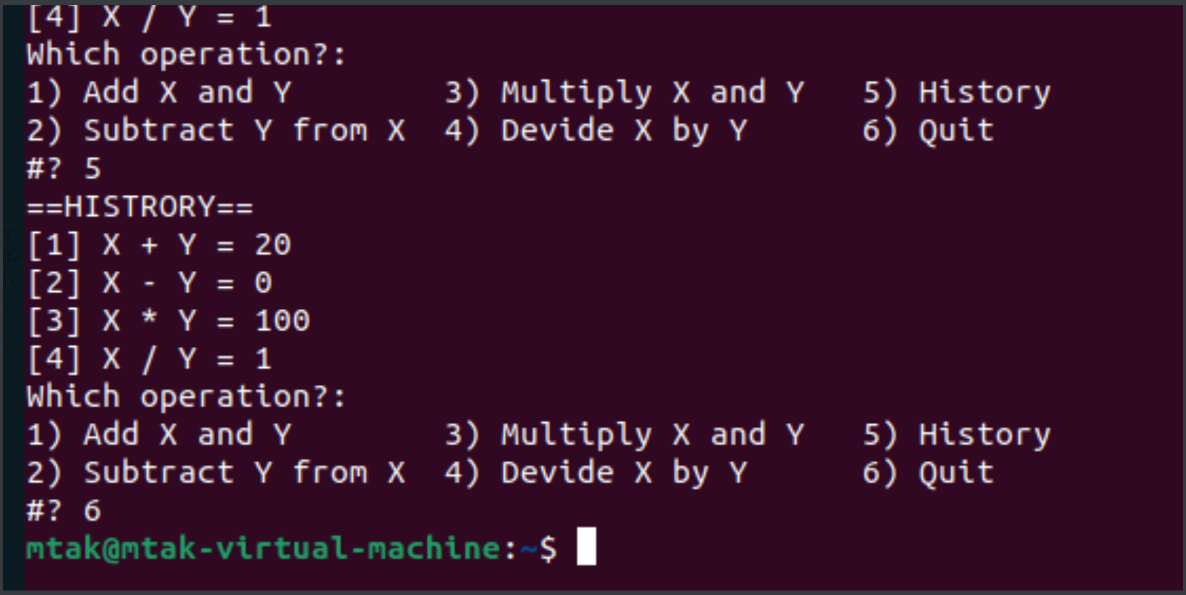
```bash
#!/bin/bash
read_x_y()
{
 echo "X?"
 read X
 echo "Y?"
 read Y
}
op=0
echo "==HISTRORY==" > ./testresults
echo "== Simple Calculator =="
while [ "$out" != "y" ]
```

```
do
 echo "Which operation?:"
 select var in "Add X and Y" "Subtract Y from X" "Multiply X and Y" "Devide X by Y" "History" "Quit"
 do
 if [ "$var" = "Add X and Y" ]
 then
   read_x_y
   op=$(($op + 1))
   echo "[$op] X + Y = $(($X + $Y))"
   echo "[$op] X + Y = $(($X + $Y))" >> ./testresults
 fi
 if [ "$var" = "Subtract Y from X" ]
 then
   read_x_y
   op=$(($op + 1))
   echo "[$op] X - Y = $(($X - $Y))"
   echo "[$op] X - Y = $(($X - $Y))" >> ./testresults
 fi
 if [ "$var" = "Multiply X and Y" ]
 then
   read_x_y
   op=$(($op + 1))
   echo "[$op] X * Y = $(($X * $Y))"
   echo "[$op] X * Y = $(($X * $Y))" >> ./testresults
 fi
 if [ "$var" = "Devide X by Y" ]
 then
   read_x_y
   op=$(($op + 1))
   echo "[$op] X / Y = $(($X / $Y))"
   echo "[$op] X / Y = $(($X / $Y))" >> ./testresults
 fi
 if [ "$var" = "Quit" ]
 then
   out="y";
```

```
  fi
  if [ "$var" = "History" ]
  then
    cat ./testresults;
  fi
  break;
  done
done

```



```
mtak@mtak-virtual-machine:~$ ./calc.sh
== Simple Calculator ==
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X  4) Devide X by Y      6) Quit
#? 1
X?
10
Y?
10
[1] X + Y = 20
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X  4) Devide X by Y      6) Quit
#? 2
X?
10
Y?
10
[2] X - Y = 0
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X  4) Devide X by Y      6) Quit
#? 3
X?
10
Y?
10
[3] X * Y = 100
Which operation?:
1) Add X and Y         3) Multiply X and Y   5) History
2) Subtract Y from X  4) Devide X by Y      6) Quit
#? 4
X?
10
Y?
10
```



```
[4] X / Y = 1
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X   4) Devide X by Y        6) Quit
#? 5
==HISTRORY==
[1] X + Y = 20
[2] X - Y = 0
[3] X * Y = 100
[4] X / Y = 1
Which operation?:
1) Add X and Y          3) Multiply X and Y    5) History
2) Subtract Y from X   4) Devide X by Y        6) Quit
#? 6
mtak@mtak-virtual-machine:~$
```