1.

```
s21900706@walab-HGU:~$ ls -l /usr/bin > ls-output.txt
```

ls 커맨드 결과를 /usr/bin 디렉토리 안에 있는 ls-output.txt 로 이동

```
s21900706@walab-HGU:~$ ls -l ls-output.txt
-rw-rw-r-- 1 s21900706 s21900706 107193 Mar 30 21:53 ls-output.txt
```

만든 ls-output.txt 의 대한 상세 내용 출력

```
s21900706@walab-HGU:~$ less ls-output.txt
```

Ls-output.txt 안에 ls 커맨드 결과물들 확인

```
s21900706@walab-HGU:~$ ls -l /bin/usr > ls-output.txt
ls: cannot access '/bin/usr': No such file or directory
```

ls 커맨드 결과를 /bin/usr 디렉토리 안에 있는 ls-output.txt 로 이동시키나 없는 디렉토리라 에러 발생

```
s21900706@walab-HGU:~$ ls -l ls-output.txt
-rw-rw-r-- 1 s21900706 s21900706 0 Mar 30 22:04 ls-output.txt
```

ls-output.txt 크기가 0 줄인걸 확인 (> 리다이렉트 시 해당 파일은 무조건 재작성 되기 때문. 재작성 시작 후 바로 에러가 발생)

```
s21900706@walab-HGU:~$ ls -l /usr/bin >> ls-output.txt
```

>>의 경우 재작성이 아닌 이어서 작성을 한다

```
s21900706@walab-HGU:~$ ls -l /usr/bin >> ls-output.txt
s21900706@walab-HGU:~$ ls -l /usr/bin >> ls-output.txt
s21900706@walab-HGU:~$ ls -l /usr/bin >> ls-output.txt
s21900706@walab-HGU:~$ ls -l ls-output.txt
-rw-rw-r-- 1 s21900706 s21900706 428772 Mar 30 22:09 ls-output.txt
```

위의 경우 여러번 리다이렉트 했기때문에 파일 크기가 배가 된걸 확인 가능

```
s21900706@walab-HGU:~$ ls -l /bin/usr 2> ls-error.txt
```

2 를 사용해 스탠다드 에러를 리다이렉트한다.

```
s21900706@walab-HGU:~$ ls -l /bin/usr > ls-error.txt 2>&1
```

기본 결과를 ls-error.txt 로 리다이렉트한 후 스탠다드 에러를 스탠다드 아웃풋으로 리다이렉트한다

```
s21900706@walab-HGU:~$ ls -l /bin/usr &> ls-output.txt
```

ls-output 파일에 &>를 이용 스탠다드 아웃풋과 스탠다드 에러를 리다이렉트한다 (재작성)

```
s21900706@walab-HGU:~$ ls -l /bin/usr &>> ls-output.txt
```

ls-output 파일에 &>를 이용 스탠다드 아웃풋과 스탠다드 에러를 리다이렉트한다 (파일에 이어서 작성한다)

```
s21900706@walab-HGU:~$ ls -l /bin/usr 2> /dev/null
```

커맨드의 결과를 사용하지 않고 버리고 싶을 때 /dev/null 파일은 bit bucket 이라 불리는 시스템 기기로 인풋값을 받고 아무것도 하지 않는다.

```
s21900706@walab-HGU:~$ cat ls-output.txt
```

ls-output 파일을 바로 디스플레이한다.

```
s21900706@walab-HGU:~$ cat
heelo world
heelo world
```

스탠다드 인풋을 받아 이를 출력하기에 키보드로 작성한것을 바로 출력한다.

```
s21900706@walab-HGU:~$ cat > lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Lazy_dog 파일에 cat 명령어 출력물을 입력한다

```
s21900706@walab-HGU:~$ cat lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Lazy_dog 파일을 불러 출력한다.

```
s21900706@walab-HGU:~$ cat < lazy_dog.txt
The quick brown fox jumped over the lazy dog.
```

Lazy_dog 파일을 입력값으로 cat 명령어를 실행한다

```
s21900706@walab-HGU:~$ ls -l /usr/bin | less
```

/usr/bing 에 대한 ls 명령어의 결과물을 less 를 이용하여 출력한다

```
s21900706@walab-HGU:~$ ls /bin /usr/bin | sort | less
```

/bin 과 /usr/bin 에 대한 ls 결과물을 sort 를 걸쳐 정리하고 less 를 통해 출력한다.

```
[s21900706@walab-HGU:~$ ls /bin /usr/bin | sort | uniq | less
```

/bin 과 /usr/bin 에 대한 ls 결과물을 sort 를 걸쳐 정리하고 uniq 를 통해 중복되는 부분들을 list 에서 지우고 less 를 통해 출력한다.

```
[s21900706@walab-HGU:~$ ls /bin /usr/bin | sort | uniq -d | less
```

/bin 과 /usr/bin 에 대한 ls 결과물을 sort 를 걸쳐 정리하고 uniq -d 를 통해 중복되는 부분만 less 를 통해 출력한다.

```
[s21900706@walab-HGU:~$ wc ls-output.txt
   2  18 112 ls-output.txt
```

ls-output 파일의 라인의 개수, 단어의 개수, 파이트의 개수를 출력한다.(2, 18, 112)

```
s21900706@walab-HGU:~$ ls /bin /usr/bin | sort | uniq | wc -l
1871
```

/bin 과 /usr/bin 에 대한 ls 결과물을 sort 를 걸쳐 정리하고 uniq 를 통해 중복자료를 삭제 후 wc -l 을 이용해 list 의 라인수를 출력한다

```
s21900706@walab-HGU:~$ ls /bin /usr/bin | sort | uniq | grep zip
bunzip2
bzip2
bzip2recover
funzip
gpg-zip
gunzip
gzip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
```

/bin 과 /usr/bin 에 대한 ls 결과물을 sort 를 걸처 정리하고 uniq 로 중복자료를 지우고 grep zip 으로 zip 이 들어간 라인을 출력한다

```
s21900706@walab-HGU:~$ head -n 5 ls-output.txt
ls: cannot access '/bin/usr': No such file or directory
ls: cannot access '/bin/usr': No such file or directory
s21900706@walab-HGU:~$ tail -n 5 ls-output.txt
ls: cannot access '/bin/usr': No such file or directory
ls: cannot access '/bin/usr': No such file or directory
```

Head -n 5 으로 ls-output 파일의 첫 5 라인을 출력하는 커맨드와 tail -n 5 으로 ls-output 파일의 마지막 5 라인을 출력하는 커맨드이다

```
s21900706@walab-HGU:~$ ls /usr/bin | tail -n 5
zipinfo
zipnote
zipsplit
zjsdecode
zlib-flate
```

Pipeline 으로도 tail 출력 가능하다(마지막 5 줄을 출력한 모습)

```
s21900706@walab-HGU:~$ tail -f /var/log/messages
tail: cannot open '/var/log/messages' for reading: No such file or directory
tail: no files remaining
```

Tail -f 옵션으로 /var/log 의 messages 파일을 계속 모니터링하고 새 라인이 써질때마다 출력한다(ctrl + c 전까지)

```
s21900706@walab-HGU:~$ ls /usr/bin | tee ls.txt | grep zip
funzip
gpg-zip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
```

/usr/bin 을 ls 한 결과를 tee 를 통해 ls.txt 파일에 쓰고 grep zip 으로 zip 이 들어간 라인을 화면으로 출력한다

```
s21900706@walab-HGU:~$ echo this is a test
this is a test
```

Echo 로 텍스트를 화면으로 출력한다

```
s21900706@walab-HGU:~$ echo *
2022OSS examples.desktop html lazy_dog.txt ls-error.txt ls-output.txt ls.txt oss
l_lab5 test.txt webapps
```

*의 뜻이 모든 파일이름을 매치하라는 뜻이기에 *가 먼저 모든 파일이름을 가져오고 echo 가 나중에 실행되어 그 내용을 출력한다

```
s21900706@walab-HGU:~$ ls
2022OSS           html          ls-error.txt   ls.txt      test.txt
examples.desktop  lazy_dog.txt  ls-output.txt  ossl_lab5   webapps
s21900706@walab-HGU:~$ echo l*
lazy_dog.txt ls-error.txt ls-output.txt ls.txt
```

현재 디렉토리에 있는 파일과 디렉토리들 중 l 이 들어가는 이름의 파일, 디렉토리들을 출력한다

```
s21900706@walab-HGU:~$ echo [[:lower:]]*
examples.desktop html lazy_dog.txt ls-error.txt ls-output.txt ls.txt ossl_lab5 t
est.txt webapps
```

현재 디렉토리에 있는 파일과 디렉토리들 중 소문자가 들어가는 이름의 파일, 디렉토리들을 다 출력한다

```
s21900706@walab-HGU:~$ echo /usr/*/share
/usr/local/share
```

현재 디렉토리 뿐만 아니라 디렉토리 이름에 /usr/*/share 형식인 디렉토리 이름도 출력한다

```
s21900706@walab-HGU:~$ echo ~
/home2/class2021/s21900706
```

홈 디렉토리의 위치를 루트 디렉토리 기준으로 출력한다

```
s21900706@walab-HGU:~$ echo ~foo
```

홈 디렉토리에 있는 foo 디렉토리의 위치를 출력한다.

```
s21900706@walab-HGU:~$ echo $((2+2))
4
```

$((표현으로)) 수학적 계산을 표현하고 2+2 의 결과를 표출한다

```
s21900706@walab-HGU:~$ echo $(($((5**2))*3))
75
```

5 의 2 승을 하고 3 을 곱한 결과를 출력한다

```
s21900706@walab-HGU:~$ echo $(((5**2)*3))
75
```

한번의 괄호로도 해당 수학적 계산이 가능하다

```
s21900706@walab-HGU:~$ echo Five divided by two equals $((5/2))
Five divided by two equals 2
s21900706@walab-HGU:~$ echo with $((5%2)) left over.
with 1 left over.
```

문장 라인과 수학적 계산을 같이 사용하여 출력 할 수 있다

```
s21900706@walab-HGU:~$ echo Front-{A,B,C}-back
Front-A-back Front-B-back Front-C-back
s21900706@walab-HGU:~$ echo number_{1..5}
number_1 number_2 number_3 number_4 number_5
```

대괄호로 여러개의 문자열을 패턴있게 출력가능하다

```
s21900706@walab-HGU:~$ echo {01..15}
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
s21900706@walab-HGU:~$ echo {001..15}
001 002 003 004 005 006 007 008 009 010 011 012 013 014 015
```

정수를 zero-padded 하여 표현도 가능하다

```
s21900706@walab-HGU:~$ echo {Z..A}
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
```

문자열과 숫자를 역순으로도 표현이 가능하다

```
s21900706@walab-HGU:~$ echo a{A{1,2},B{3,4}}b
aA1b aA2b aB3b aB4b
```

Brace 안에 brace 를 연속으로 사용이 가능하다

```
s21900706@walab-HGU:~$ mkdir Photos
s21900706@walab-HGU:~$ cd Photos/
s21900706@walab-HGU:~/Photos$ mkdir {2007..2009}-{01..12}
s21900706@walab-HGU:~/Photos$ ls
2007-01  2007-05  2007-09  2008-01  2008-05  2008-09  2009-01  2009-05  2009-09
2007-02  2007-06  2007-10  2008-02  2008-06  2008-10  2009-02  2009-06  2009-10
2007-03  2007-07  2007-11  2008-03  2008-07  2008-11  2009-03  2009-07  2009-11
2007-04  2007-08  2007-12  2008-04  2008-08  2008-12  2009-04  2009-08  2009-12
```

Photos 라는 디렉토리 생성 후 해당 디렉토리에 brace 를 이용한 연속 디렉토리를 생성

```
s21900706@walab-HGU:~$ echo $USER
s21900706
```

유저의 유저이름을 출력

```
s21900706@walab-HGU:~$ printenv | less
```

사용가능한 변수들을 모두 볼 수 있게 함

```
s21900706@walab-HGU:~$ echo $SUER

```

변수의 이름을 잘못 적을 경우 빈칸을 출력한다

```
s21900706@walab-HGU:~$ echo $(ls)
2022OSS examples.desktop html lazy_dog.txt ls-error.txt ls-output.txt ls.txt oss
l_lab5 test.txt webapps
```

ls 의 내용을 에코로 출력한다

```
[s21900706@walab-HGU:~$ ls -l $(which cp)
-rwxr-xr-x 1 root root 141528 Jan 18  2018 /bin/cp
```

Cp 라는 이름의 디렉토리에 대한 상세 내용을 자세한 경로이름을 몰라도 출력 가능하다

```
[s21900706@walab-HGU:~$ file $(ls -d /usr/bin/* | grep zip)
/usr/bin/funzip:     ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dyn
amically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, B
uildID[sha1]=7aa1524b9c858c29db1dba7fa04ae27cdb6c5de4, stripped
/usr/bin/gpg-zip:    POSIX shell script, ASCII text executable
/usr/bin/mzip:       symbolic link to mtools
/usr/bin/preunzip:   POSIX shell script, ASCII text executable
/usr/bin/prezip:     POSIX shell script, ASCII text executable
/usr/bin/prezip-bin: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dyn
amically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, B
uildID[sha1]=d4270e25323dcb27d324eabca9dd8d33be2a804c, stripped
/usr/bin/unzip:      ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dyn
amically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, B
uildID[sha1]=ac1eb2744561bfb2319b99ece51815b5caab0dfd, stripped
/usr/bin/unzipsfx:   ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dyn
amically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, B
```

/usr/bin 내 디렉토리중 grep zip 으로 zip 이 들어가는 파일을 file 으로 표현한다

```
[s21900706@walab-HGU:~$ echo this is a      test
this is a test
```

에코에서 단어 사이 과도한 빈칸들을 지운다

```
[s21900706@walab-HGU:~$ echo The total is $100.00
The total is 00.00
```

$1 이 알 수 없는 변수라 해당변수를 무시하고 출력한다

```
[s21900706@walab-HGU:~$ ls -l two words.txt
ls: cannot access 'two': No such file or directory
```

두 단어 사이 공백으로 two 와 words.txt 를 따로 판단하여 제대로 된 기능 수행이 불가

```
[s21900706@walab-HGU:~$ ls -l "two words.txt"
ls: cannot access 'two words.txt': No such file or directory
[s21900706@walab-HGU:~$ mv "two words.txt" two_words.txt
mv: cannot stat 'two words.txt': No such file or directory
```

“”으로 한 파일이라는걸 알려 줄 수 있고 파일이름을 바꿔 더 쉽게 접근가능하게 바꿈

```
[s21900706@walab-HGU:~$ echo "$USER $((2+2)) $(cal)"
s21900706 4      March 2022
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

“”으로 하나도 인식하게 하여 전체 출력한다

```
[s21900706@walab-HGU:~$ echo this is a      test
this is a test
[s21900706@walab-HGU:~$ echo "this is a      test"
this is a      test
```

“”사용 시 하나로 인식한다

```
[s21900706@walab-HGU:~$ echo $(cal)
March 2022 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31
[s21900706@walab-HGU:~$ echo "$(cal)"
     March 2022
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
```

Cal 의 결과물을 에코로 출력하는 것과 cal 함수자체를 에코로 출력하는 것의 차이를 알 수 있다.

```
[s21900706@walab-HGU:~$ echo text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
text /home2/class2021/s21900706/lazy_dog.txt /home2/class2021/s21900706/ls-error
.txt /home2/class2021/s21900706/ls-output.txt /home2/class2021/s21900706/ls.txt
/home2/class2021/s21900706/test.txt a b foo 4 s21900706
```

각 빈칸에 맞춰 다른 파일로 인식해 따로 출력된 모습

```
s21900706@walab-HGU:~$ echo "text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/*.txt {a,b} foo 4 s21900706
```

“”사이를 하나의 라인으로 인식해서 문장들과 함수선언들의 결과를 출력

```
[s21900706@walab-HGU:~$ echo 'text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/*.txt {a,b} $(echo foo) $((2+2)) $USER
```

‘’사이 자체를 문장으로만 취급하여 그대로 출력

```
s21900706@walab-HGU:~$ echo "The balance for user $USER is : \$5.00"
The balance for user s21900706 is : $5.00
```

\로 함수결과물을 함수가 아닌 문자로 취급함

```
s21900706@walab-HGU:~$ ls
2022OSS           html          ls-error.txt   ls.txt      test.txt
examples.desktop  lazy_dog.txt  ls-output.txt  ossl_lab5   webapps
s21900706@walab-HGU:~$ ls l
lazy_dog.txt   ls-error.txt   ls-output.txt  ls.txt
```

Tab 으로 l 로 시작하는 파일들을 보여주는 모습(파일이 하나면 자동으로 그 파일이름을 가져와 준다

```
s21900706@walab-HGU:~$ history | less
```

이제까지 터미널에 사용했던 커맨드들을 보여줌

```
s21900706@walab-HGU:~$ history | grep /usr/bin
   79  cd /usr/bin
   89  cd /usr/bin
   95  cd /usr/bin
  435  ls -l /usr/bin > ls-output.txt
  445  ls -l /usr/bin >> ls-output.txt
  461  ls -l /usr/bin | less
  462  ls /bin /usr/bin | sort | less
  463  ls /bin /usr/bin | sort | uniq | less
  464  ls /bin /usr/bin | sort | uniq -d | less
  470  ls /bin /usr/bin | sort | uniq | wc -l
  471  ls /bin /usr/bin | sort | uniq | grep zip
  474  ls /usr/bin | tail -n 5
  476  ls /usr/bin | tee ls.txt | grep zip
  510  history | grep /usr/bin
```

이제까지 사용했던 커맨드중 /usr/bin 이 들어가는 커맨드들을 출력

```
s21900706@walab-HGU:~$ !88
cd ../..
```

88 번째 사용했던 커맨드 출력

```
(reverse-i-search)`/usr/bin': cd /usr/bin
```

바로 /usr/bin 이 들어갔던 최근 커맨드를 불러와서 실행 가능

```
s21900706@walab-HGU:~$ ls -l /usr/bin > ls-output.txt
```

/usr/bin 에 있는 자료들에 대한 내용을 ls-output.txt 파일에 복사

파일 접근 권한
R : 파일을 read 할 수 있는 권한
W : 파일을 변환 할 수 있는 권한
X : 파일을 실행할 수 있는 권한

- : 일반 파일을 의미하는 시작
d : 디렉토리를 의미하는 시작

chmod : 파일의 접근권한을 바꿀 때 사용 777 등

3 가지의 적용점
1. 파일의 원 소유주 (u 로 표현)
2. 파일 소유 그룹원 (g 로 효현)
3. 기타 (o 로 표현)
전부는 a 로 표현

예시
u+x : 유저에게 실행 권한을 추가
u-x : 유저에게서 실행 권한 삭제
go=rw : 소유 그룹원과 기타원들에 대해 읽고 쓸 권한을 줌

umask : 내가 만드는 파일들의 기본 권한 설정을 바꿀 때 사용

su : 현재 사용자에서 다른 사용자로 바꿀 때 사용
su - : 슈퍼유저로 접속할 때

sudo : 관리자가 일반 유저에게 특별 권한을 부여 할 때 사용
슈퍼 유저의 비밀번호가 아닌 자신의 비밀번호로 접근 가능

shown : 파일의 소유주와 그룹 오너들을 바꿀 때 사용 (슈퍼유저 권한일 때 가능)

이름 : 파일 소유주를 이름으로 변경

이름:이름 1 : 파일의 소유주를 이름으로 바꾸고 파일 그룹 소유주를 이름 1 로 변경
:이름 : 그룹 소유주를 이름으로 변경
이름: : 파일 소유주, 그룹 소유주, 기타의 권한을 모두 이름으로 변경

passwd : 현재 사용자의 비밀번호를 변경

ps : 현재 진행중인 프로세스를 보여준다
top : 실시간으로 시스템 프로세스/메모리 사용 현황을 보여준다
jobs : 작업이 중지된 상태나 백그라운드로 진행 중인 상태를 표시
bg : 백그라운드 실행
fg : 포어그라운드 실행
kill : 프로세스에 종료 신호를 보냄
killall : 프로세스 이름을 이용해 직접 종료함(kill -PID 값)
shutdown : 시스템을 안전하게 종료하는 시스템 관리 명령어(관리자만 사용가능)

2)

#. 문제 번호

a) 사용된 쉘 명령어

b) test.c 파일 변경내용

c) vim 으로 수행하기 위해 필요한 키보드 키

1.
a) mkdir 2022OSS, cd 2022OSS

2.
a) mkdir lab05, cd lab05

3.
a) ln -s 2022OSS/lab5 ossl_lab5

4.
a) cd ossl_lab5

5.
a) vim test.c
c) i

6.
b) #include <stdio.h>
int main(){
 return 0;
}

7.
c) ZZ

8.
a) vim test.c

9.
c) :set nu

10.
c) O

11.

c) enter key pressed at the end of the line

12.
b) #include <stdio.h>

```
int main(){
        printf(“Hello World!!!\n”);
         return 0;
}
```

13.
c) ZZ

14.
a) gcc -o test test.c

15.
a) ls

16.
a)./test

17.
a) vim test.c

18.
c) kk

19.
c) yy

20.
b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4       printf("Hello World!!!\n");
 5       printf(“Hello World!!!\n");
 6.      printf(“Hello World!!!\n");
 7       return 0;
 8 }
```

c) pp

21.

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4       printf("Hello World!!!\n");
 5       printf(“Hello World!!!\n");
 6       return 0;
 7 }
```

c) u

22.

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4       printf("Hello World!!!\n");
 5       printf(“\n");
 6       return 0;
 7 }
```

c) dwdwxx

23.

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4       printf("Hello World!!!\n");
 5       printf(“Name : 나한동\n");
 6       return 0;
 7 }
```

c) I Name : 나한동 esc ZZ

24.

a) gcc -o test test.c

25.

a) ./test

26.

a) vim test.c

27.

a) gcc -o test test.c ./test

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4        printf("Shalom!!!\n");
 5        printf(“Name : 나한동\n");
 6        return 0;
 7 }
```

c) dwdw Shalom ZZ

28.

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4        printf("Shalom!!!\n");
 5        printf("Name : 조영관\n");
 6       return 0;
 7 }
```

29.

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4        printf("Shalom!!!\n");
 5        printf("Name : 조영관\n");
 6        printf("In his heart a man plans his course, but the LORD determines     his steps.
(Proverbs 16:9)\n");
 7        return 0;
 8 }
```

30.

a) gcc -o test test.c ./test

b)

```
 1 #include <stdio.h>
 2
 3 int main(){
 4        printf("Shalom!!!\n");
 5        printf("Name : 조영관\n");
 6        printf("In his heart a man plans his course, but the LORD determines     his steps.
(Proverbs 16:9)\n");
```

```
7       return 0;
8 }
```
c) ZZ