<인하대 오픈소스 SW 개론 Bash 과제>

과목 : 오픈소스SW개론
학과 : 인공지능공학과
학번 : 12215332
이름 : 팽준호

스크립트 커맨드 : bash prj1_12215332_paengjunho.sh u.item u.data u.user

#! /bin/bash -> 스크립트가 Bash를 사용하여 실행됨.

```
# 스크립트 실행 시 전달된 인수를 변수에 할당함.
# $1, $2, $3은 스크립트를 실행할 때, 전달된 인수를 나타냄.
# $1, $2, $3 <- u.item, u.data, u.user를 나타냄.
item=$1
data=$2
user=$3

# 과제 pdf의 형식을 echo를 사용하여 나타냄.
echo "--------------------------"
echo "User Name: $(whoami)"
echo "Student Number: 12215332"
echo "[ MENU ]"
echo "1. Get the data of the movie identified by a specific 'movie id' from 'u.item'"
echo "2. Get the data of action genre movies from 'u.item'"
echo "3. Get the average 'rating' of the movie identified by specific 'movie id' from 'u.data'"
echo "4. Delete the 'IMDb URL' from 'u.item'"
echo "5. Get the data about users from 'u.user'"
echo "6. Modify the format of 'release date' in 'u.item'"
echo "7. Get the data of movies rated by a specific 'user id' from 'u.data'"
echo "8. Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'"
echo "9. Exit"
echo "--------------------------"
```

-> 밑에 그림을 보시면 과제 pdf와 제가 작성한 shell 스크립트의 결과값을 비교해 보겠습니다.

(과제 pdf)

```
--------------------------
User Name: fos
Student Number: 00000000
[ MENU ]
1. Get the data of the movie identified by a specific
'movie id' from 'u.item'
2. Get the data of action genre movies from 'u.item'
3. Get the average 'rating' of the movie identified by
specific 'movie id' from 'u.data'
4. Delete the 'IMDb URL' from 'u.item
5. Get the data about users from 'u.user'
6. Modify the format of 'release date' in 'u.item'
7. Get the data of movies rated by a specific 'user id'
from 'u.data'
8. Get the average 'rating' of movies rated by users with
'age' between 20 and 29 and 'occupation' as 'programmer'
9. Exit
--------------------------
```

whoami

(제가 작성한 쉘 스크립트의 결과값)

```
ubuntu@ubuntu-VirtualBox:~$ bash test.sh u.item u.data u.user
----------------------
User Name: ubuntu
Student Number: 12215332
[ MENU ]
1. Get the data of the movie identified by a specific 'movie id' from 'u.item'
2. Get the data of action genre movies from 'u.item'
3. Get the average 'rating' of the movie identified by specific 'movie id' from 'u.data'
4. Delete the 'IMDb URL' from 'u.item'
5. Get the data about users from 'u.user'
6. Modify the format of 'release date' in 'u.item'
7. Get the data of movies rated by a specific 'user id' from 'u.data'
8. Get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as
 'programmer'
9. Exit
----------------------
```

-> 과제 유형에 맞게 똑같이 출력되는 것을 보실 수 있습니다.

무한 loop를 while문을 통해 실행
-> 무한 루프를 통해 사용자의 선택을 반복적으로 처리합니다.

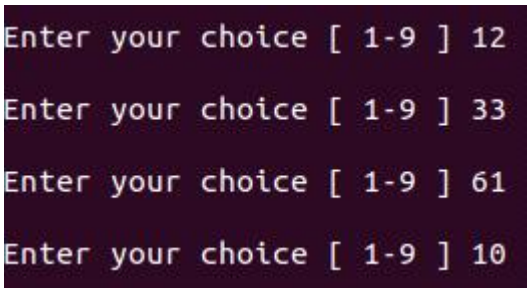
```
while true
do
    echo
    read -p "Enter your choice [ 1-9 ] " choice
```

-> 변수 choice에 1 ~ 9 사이 숫자를 저장하도록 합니다.

(과제 pdf)

```
Enter your choice [ 1-9 ] 3
```

(제가 작성한 쉘 스크립트의 결과값)



-> 여기서 1 ~ 9번 사이의 값이 아닐 경우 계속해서 반복문이 돌아가면서 “Enter your choice [1-9] ” 이 문장을 나오도록 continue를 이용했습니다,

사용자가 메뉴 1번을 선택함 -> 1번 : 사용자가 1~1682 사이에 있는 번호를 입력하면 u.item에 있는 파일안에서 movie id와 같은 번호를 매칭하여 그 줄을 출력함.

```
if [ $choice = 1 ]; then
    echo
    read -p "Please enter 'movie id' (1~1682):" movie_id
    awk -v line="$movie_id" 'NR==line' $item
```

<1번 알고리즘>

1. 변수 movie_id에 입력을 받습니다.
2. u.item 파일안에 외부 변수 movie_id을 line으로 넣습니다.
3. line과 일치하는 행을 출력합니다.

<1번 코드 설명>

-> 변수 choice가 1이라면 "Please enter 'movie id' (1~1682):" 이 문자을 출력하고 1에서 1682사이의 숫자를 입력하게 합니다.

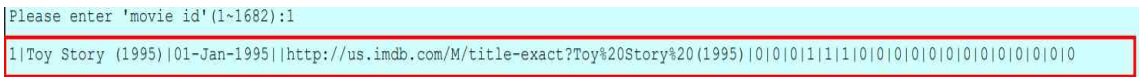
-> awk 명령어를 이용하여 특정 조건을 만족하는 파일의 레코드 라인을 선택하여 출력합니다.

-> -v는 awk 명령어에서 사용되는 옵션으로 awk 스크립트 내에서 외부 변수를 사용할 수 있으며, 이를 통해 외부 데이터를 스크립트로 가져올 수 있습니다.

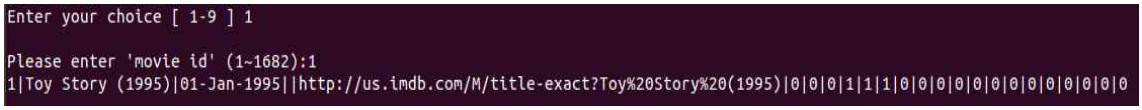
-> NR은 awk에서 현재 처리 중인 레코드의 번호를 나타내며, line은 사용자가 입력한 외부 변수(movie_id)와 일치하는 레코드(행)를 찾는 패턴입니다.

-> 즉, awk를 써서 u.item파일을 읽어오고, -v를 이용해 외부 변수 movie_id를 가져와서 NR과 외부 변수 movie_id가 일치하면 u.item 파일의 해당되는 줄(행)을 출력합니다.

(과제 1번 pdf)



(제가 작성한 쉘 스크립트의 1번 결과값)



사용자가 메뉴 2번을 선택함 -> 2번 : 사용자가 의사표시를 yes로 표현하면 u.item 파일에서 장르가 action에 해당하는 $1(영화 ID)와 $2(영화 제목)을 출력함.
사용자로부터 액션 장르 영화 데이터를 출력할 지 묻고, 사용자가 'y'를 선택한 경우 해당 데이터를 출력함.

```
    elif [ $choice = 2 ]; then
        echo
        read -p "Do you want to get the data of 'action' genre movies from 'u.item'? (y/n):" intention
        if [ $intention = "y" ]; then
            echo
            awk -F'|' '$7 {print $1, $2}' $item | sort -n | head -n 10
        else
            continue
        fi
```

<2번 알고리즘>

1. u.item 파일 안에 $7(action)이면 그 줄에 해당하는 영화 ID와 영화 제목을 출력합니다.

<2번 코드 설명>

-> $intention은 yes/no 의사표시를 저장할 변수입니다. $intention가 ‘y’라면 메뉴 2번에 대한 내용을 출력하고 ‘y’가 아니면 "Enter your choice [1-9] "로 돌아가도록 설계했습니다.

-> awk로 u.item 파일을 읽어들이고 -F로 구분자‘|’로 하면서 u.item 파일의 열들을 읽어들입니다. 그러면 $1 = 영화 ID, $2 = 영화 제목이 됩니다.

-> '$7 {print $1, $2}' : $7 = action장르이기 때문에 $7이 있으면 그에 해당하는 $1과 $2를 출력하도록 만들었습니다.

-> sort -n : sort 명령어를 사용하여 오름차순 시켜줍니다.

-> head -n 10 : head 명령어를 사용하여 처음(상위) 10개의 행만 출력합니다.

-> 즉, u.item안에 있는 $7(action genre)에 해당하는 영화 ID와 영화 제목을 영화 ID를 기준으로 오름차순 정렬하고 그 중 위의 10개를 출력합니다.

(과제 2번 pdf)

```
Enter your choice [ 1-9 ] 2

Do you want to get the data of 'action' genre movies
from 'u.item'?(y/n):y

2 GoldenEye (1995)
4 Get Shorty (1995)
17 From Dusk Till Dawn (1996)
21 Muppet Treasure Island (1996)
22 Braveheart (1995)
24 Rumble in the Bronx (1995)
27 Bad Boys (1995)
28 Apollo 13 (1995)
29 Batman Forever (1995)
33 Desperado (1995)

Enter your choice [ 1-9 ]
```

(제가 작성한 쉘 스크립트의 2번 결과값)

```
Enter your choice [ 1-9 ] 2

Do you want to get the data of 'action' genre movies from 'u.item'? (y/n):y

2 GoldenEye (1995)
4 Get Shorty (1995)
17 From Dusk Till Dawn (1996)
21 Muppet Treasure Island (1996)
22 Braveheart (1995)
24 Rumble in the Bronx (1995)
27 Bad Boys (1995)
28 Apollo 13 (1995)
29 Batman Forever (1995)
33 Desperado (1995)

Enter your choice [ 1-9 ]
```

사용자가 메뉴 3번을 선택함 -> 3번 : 사용자가 영화 ID를 입력하면 u.data 파일에 있는 열을 읽어서 평균 평점을 구하도록 함.

```
elif [ $choice = 3 ]; then
    echo
    read -p "Please enter the 'movie id' (1~1682):" input_movie_id
    count=0
    rating_sum=0
    while read -r user_id movie_id rating timestamp; do
        if [ "$movie_id" -eq "$input_movie_id" ]; then
            count=$((count + 1))
            rating_sum=$((rating_sum + rating))
        fi
    done < $data
    result=$(echo "scale=6; $rating_sum / $count" | bc)
    result=$(printf "%.5f" $result)
    echo
    echo "average rating of $input_movie_id: $result“
```

<3번 알고리즘>

1. 반복문을 돌려서 입력한 영화 ID에 일치하면 평점 총합과 count를 계속 더해줍니다.
2. 반복문이 끝나면 평균을 냅니다.
3. 평균 결과값의 소수점 6번째 자리에서 반올림하여 소수점 5번째 자리까지만 출력하도록 합니다.

<3번 코드 설명>

-> 반복문을 통해 u.data의 필드(열)마다 열들의 정보를 읽어서 변수 4개에 저장합니다.

-> u.data로 읽어들인 영화 ID와 $input_movie_id가 같으면(-eq) count 변수를 1씩 증가시키고 평점을 변수 rating_sum에 계속 더합니다.

-> echo 명령어를 사용하여 bc에게 부동 소수점 연산을 수행하도록 합니다.

-> scale=6은 bc에게 소수점 아래 6자리까지 결과를 표시하도록 지시합니다.

-> "%.5f"는 소수점 아래 5자리까지 표시하도록 지정하는 서식 지정자입니다 이것은 알아서 소수 6번째 자리를 반올림하여 소수점 5번째 자리까지 표현합니다.

(과제 3번 pdf)

```
Enter your choice [ 1-9 ] 3
Please enter the 'movie id'(1~1682):1
average rating of 1: 3.87832
```

(제가 작성한 쉘 스크립트의 3번 결과값)

```
Enter your choice [ 1-9 ] 3
Please enter the 'movie id' (1~1682):1
average rating of 1: 3.87832
```

사용자가 메뉴 4번을 선택함 -> 4번 : u.item 파일에서 $5(IMDb URL)를 제거하고 상위 10개를 출력함.

```
    elif [ $choice = 4 ]; then
        echo
        read -p "Do you want to delete the 'IMDb URL' from 'u.item'? (y/n):" intention
        if [ $intention = "y" ]; then
            echo
            awk -F'|' '{ for(i = 1; i <= NF; i++) {
                if(i != 5) { printf "%s",$i; if(i < NF) printf "|" }
                else { printf "|" }} print"" }' $item | head -n 10
        else
            continue
        fi
```

<4번 알고리즘>

1. 반복문을 통해 열들을 읽어서 IMDb URL의 정보이면 출력하지 않습니다.

<4번 코드 설명>

-> awk -F'|' : 구분자 '|'를 제거하면서 파일의 필드를 읽고 출력합니다.

-> NF는 awk 스크립트에서 사용되는 내장 변수로, 현재 레코드(행)의 필드(열) 수를 나타냅니다.

-> $5가 MDb URL이기 때문에 for문을 이용하여 $5라면 $5의 정보가 아니라 '|'을 출력하도록 만들었습니다.

-> 즉, $5인 MDb URL의 필드를 제외한 각 필드마다 정보를 출력하고 사이 사이에 '|'을 출력하고 i가 $5라면 $5의 정보 대신 '|'을 출력하도록 만들었습니다.

(과제 4번 pdf)

```
Enter your choice [ 1-9 ] 4

Do you want to delete the 'IMDb URL' from 'u.item'?(y/n):y

1|Toy Story (1995)|01-Jan-1995|||0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995|||0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995|||0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995|||0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-
1995|||0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
7|Twelve Monkeys (1995)|01-Jan-1995|||0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0
8|Babe (1995)|01-Jan-1995|||0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0|0|0
9|Dead Man Walking (1995)|01-Jan-1995|||0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
10|Richard III (1995)|22-Jan-1996|||0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0
```

(제가 작성한 쉘 스크립트의 4번 결과값)

```
Enter your choice [ 1-9 ] 4

Do you want to delete the 'IMDb URL' from 'u.item'? (y/n):y

1|Toy Story (1995)|01-Jan-1995|||0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995|||0|1|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995|||0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995|||0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995|||0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-1995|||0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
7|Twelve Monkeys (1995)|01-Jan-1995|||0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0
8|Babe (1995)|01-Jan-1995|||0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0|0|0
9|Dead Man Walking (1995)|01-Jan-1995|||0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
10|Richard III (1995)|22-Jan-1996|||0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0
```

사용자가 5번을 선택함 -> 5번 : 사용자로부터 사용자 데이터를 출력할 지 묻고, 사용자가 'y'를 선택한 경우 u.user파일 안에서 gender의 M -> male, F -> female로 바꿔서 출력함.

```
elif [ $choice = 5 ]; then
        echo
        read -p "Do you want to get the data about users from 'u.user'? (y/n):" intention
        if [ $intention = "y" ]; then
            echo
            awk -F'|' '{if($3 == "M") {$3 = "male"} else {$3 = "female"}} {print "user",$1,"is",$2,"years old",$3,$4}' $user | head -n 10
        else
            continue
        fi
```

<5번 알고리즘>

1. u.user 파일을 읽어서 gender열 부분이 M -> male, F -> female로 바꾼 다음 과제 형식에 맞게 출력합니다.

<5번 코드 설명>

-> u.user 파일 안에서 '|' 구분자들을 지우면서 필드들에 정보를 읽고 $3(gender)을 읽어들여서 M이면 mele, F이면 female로 바꿔서 형식에 맞춰서 상위 10개 출력합니다.

-> 즉, if-else문을 이용하여 u.user 파일 안에 $3(gender)가 M이면 male로 F이면 female로 바꾼 뒤 상위 10개를 과제 형식에 맞게 출력하였습니다.

(과제 5번 pdf)

```
Enter your choice [ 1-9 ] 5

Do you want to get the data about users from
'u.user'?(y/n):y

user 1 is 24 years old male technician
user 2 is 53 years old female other
user 3 is 23 years old male writer
user 4 is 24 years old male technician
user 5 is 33 years old female other
user 6 is 42 years old male executive
user 7 is 57 years old male administrator
user 8 is 36 years old male administrator
user 9 is 29 years old male student
user 10 is 53 years old male lawyer
```

(제가 작성한 쉘 스크립트의 5번 결과값)

```
Enter your choice [ 1-9 ] 5

Do you want to get the data about users from 'u.user'? (y/n):y

user 1 is 24 years old male technician
user 2 is 53 years old female other
user 3 is 23 years old male writer
user 4 is 24 years old male technician
user 5 is 33 years old female other
user 6 is 42 years old male executive
user 7 is 57 years old male administrator
user 8 is 36 years old male administrator
user 9 is 29 years old male student
user 10 is 53 years old male lawyer
```

사용자가 6번을 선택함 -> 6번 : 사용자로부터 출시 날짜 형식을 수정할 지 묻고, 사용자가 'y'를 선택한 경우, u.item에 있는 날짜 형식을 우리가 아는 날짜 형식으로 바꿔서 출력함.

```
    elif [ $choice = 6 ]; then
        echo
        read -p "Do you want to Modify the format of 'release date' in 'u.item'?(y/n):" intention
        if [ $intention = "y" ]; then
            echo
            awk -F'|' '{
                gsub("-","",$3);
                gsub("Jan","01",$3);
                gsub("Feb","02",$3);
                gsub("Mar","03",$3);
                gsub("Apr","04",$3);
                gsub("May","05",$3);
                gsub("Jun","06",$3);
                gsub("Jul","07",$3);
                gsub("Aug","08",$3);
                gsub("Sep","09",$3);
                gsub("Oct","10",$3);
                gsub("Nov","11",$3);
                gsub("Dec","12",$3);
                $3 = substr($3,5,4) substr($3,3,2) substr($3,1,2);
                for(i = 1; i <= NF; i++) { printf "%s",$i; if(i < NF) printf "|" } print ""}' $item | tail -n 10
        else
            continue
        fi
```

<6번 알고리즘>

1. u.item 파일에서 날짜 형식만 추출합니다.
2. 날짜 형식안에서 - 문자를 지웁니다.
3. 그리고 1~12월로 되어있는 영어문자를 숫자 형식으로 바꿉니다.
4. 각 문자 위치 재배치하여 기존의 날짜 형식 자리에 바꾼 값을 안에 넣습니다.
5. 과제 형식에 맞춰서 출력합니다.

<6번 코드 설명>

-> awk를 이용해 u.item 파일에서 $3(날짜 형식)을 읽고 gsub명령어를 이용해서 "-"를 없애고 각 영어로 되어있는 1 ~ 12월을 숫자 형식의 문자열로 바꿉니다.

-> substr(문자열, 시작 위치, 길이) : 특정 위치에서 시작하여 지정된 길이나 인덱스 범위 내의 문자열 부분을 추출합니다.

-> 예시 : substr() 명령어를 이용하여 01011995있던 $3를 19950101로 재배치하여 $3에 다시 넣어줍니다.

-> for문으로 각 열들의 정보를 읽어들이고 하위 10개만 출력하도록 만들었습니다.

(과제 6번 pdf)

```
Enter your choice [ 1-9 ] 6

Do you want to Modify the format of 'release data' in 'u.item'?(y/n):y

1673|Mirage (1995)|19950101|http://us.imdb.com/M/title-exact?Mirage%20(1995)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
1674|Mamma Roma (1962)|19620101||http://us.imdb.com/M/title-exact?Mamma%20Roma%20(1962)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1675|Sunchaser, The (1996)|19961025||http://us.imdb.com/M/title-exact?Sunchaser,%20The%20(1996)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1676|War at Home, The (1996)|19960101||http://us.imdb.com/M/title-exact?War%20at%20Home%2C%20The%20%281996%29|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1677|Sweet Nothing (1995)|19960920||http://us.imdb.com/M/title-exact?Sweet%20Nothing%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1678|Mat' i syn (1997)|19980206||http://us.imdb.com/M/title-exact?Mat%27+i+syn+(1997)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1679|B. Monkey (1998)|19980206||http://us.imdb.com/M/title-exact?B%2E+Monkey+(1998)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|1|0|0
1680|Sliding Doors (1998)|19980101||http://us.imdb.com/Title?Sliding+Doors+(1998)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
1681|You So Crazy (1994)|19940101||http://us.imdb.com/M/title-exact?You%20So%20Crazy%20(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
1682|Scream of Stone (Schrei aus Stein) (1991)|19960308||http://us.imdb.com/M/title-
exact?Schrei%20aus%20Stein%20(1991)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
```

(제가 작성한 쉘 스크립트의 6번 결과값)

```
Enter your choice [ 1-9 ] 6

Do you want to Modify the format of 'release data' in 'u.item'?(y/n):y

1673|Mirage (1995)|19950101||http://us.imdb.com/M/title-exact?Mirage%20(1995)|0|1|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
1674|Mamma Roma (1962)|19620101||http://us.imdb.com/M/title-exact?Mamma%20Roma%20(1962)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1675|Sunchaser, The (1996)|19961025||http://us.imdb.com/M/title-exact?Sunchaser,%20The%20(1996)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1676|War at Home, The (1996)|19960101||http://us.imdb.com/M/title-exact?War%20at%20Home%2C%20The%20%281996%29|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1677|Sweet Nothing (1995)|19960920||http://us.imdb.com/M/title-exact?Sweet%20Nothing%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1678|Mat' i syn (1997)|19980206||http://us.imdb.com/M/title-exact?Mat%27+i+syn+(1997)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
1679|B. Monkey (1998)|19980206||http://us.imdb.com/M/title-exact?B%2E+Monkey+(1998)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|1|0|0
1680|Sliding Doors (1998)|19980101||http://us.imdb.com/Title?Sliding+Doors+(1998)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|0
1681|You So Crazy (1994)|19940101||http://us.imdb.com/M/title-exact?You%20So%20Crazy%20(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
1682|Scream of Stone (Schrei aus Stein) (1991)|19960308||http://us.imdb.com/M/title-exact?Schrei%20aus%20Stein%20(1991)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
```

사용자가 7번을 선택함 -> 7번 : 사용자로부터 사용자 ID를 입력받고, 해당 사용자가 평가한 영화 ID를 출력한 뒤, 해당하는 영화 제목 상위 10개의 목록을 출력함.

```
    elif [ $choice = 7 ]; then
        echo
        read -p "Please enter the 'user id' (1~943):" input_user_id
        echo
        find_movie_id=$(awk -v user_id="$input_user_id" -F' ' '$1 == user_id {print $2}' "$data" | sort -n | tr '\n' '|' | sed 's/|$//')
        echo $find_movie_id
        echo
        IFS='|' read -ra movie_id_array <<< "$find_movie_id"
        for((i=0;i<10;i++)); do
            awk -v movie_id="${movie_id_array[i]}" -F'|' '$1 == movie_id {printf "%s|%s\n", $1,$2}' "$item"
        done
```

<7번 알고리즘>

1. 유저 아이디를 입력합니다.
2. 입력한 유저 아이디를 변수에 저장합니다.
3. 저장된 변수와 u.data 파일에서의 user id가 일치하면 영화 ID를 추출해서 1|2|3|4|5 이런 식으로 만들어서 다른 변수에 정렬하여 저장합니다.
4. 저장한 1|2|3|4|5 이런 형식의 변수를 출력합니다.
5. 저장한 변수에 대하여 |을 기준으로 영화 ID를 구분하고 배열 변수를 만듭니다.
6. 반복문을 10번만 돌려 배열 변수 원소 1개씩 u.item파일의 영화 ID와 일치하면 영화 ID와 영화 제목을 출력합니다.

<7번 코드 설명>

-> -v user_id="$input_user_id" : -v 옵션을 사용하여 awk 스크립트 내에서 변수 user_id를 외부 변수 input_user_id 값으로 설정합니다.

-> | tr '\n' '|': 결과값에서 개행 문자(\n)를 수직 바(|)로 변경합니다.

-> sed 's/|$//는 결과에서 마지막의 수직 바(|)를 삭제합니다. 그리고 awk -v user_id="$input_user_id" -F' ' '$1 == user_id {print $2}' "$data" | sort -n | tr '\n' '|' | sed 's/|$//'의 결과값을 변수 find_movie_id에 저장합니다.

-> 변수 find_movie_id의 값을 ‘|’을 구분자로 하여 정보를 분석합니다.

-> ISF '|' : Bash 스크립트에서 문자열을 필드(요소)로 분할할 때 사용되는 구분자 '|'로 설정하고 이 read를 통해 Bash는 문자열을 '|'을 기준으로 필드로 분리합니다.

-> IFS를 수직 바(|)로 설정하고, $find_movie_id을 '|'을 기준으로 필드로 분할합니다.

-> 즉, read 명령어를 사용하여 문자열($find_movie_id)을 필드로 분할하고, -ra를 사용하여 분리된 필드를 배열인 movie_id_array에 저장합니다.

-> 배열 요소 중 10개만 u.item 파일 안에서의 영화 ID와 일치하면 영화 ID와 영화 제목을 출력하도록 만들었습니다.

(과제 7번 pdf)

```
Enter your choice [ 1-9 ] 7

Please enter the 'user id'(1~943):12

4|15|28|50|69|71|82|88|96|97|98|127|132|133|143|15
7|159|161|168|170|172|174|191|195|196|200|202|203|
204|215|216|228|238|242|276|282|300|318|328|381|39
2|402|416|471|480|591|684|708|735|753|754

4|Get Shorty (1995)
15|Mr. Holland's Opus (1995)
28|Apollo 13 (1995)
50|Star Wars (1977)
69|Forrest Gump (1994)
71|Lion King, The (1994)
82|Jurassic Park (1993)
88|Sleepless in Seattle (1993)
96|Terminator 2: Judgment Day (1991)
97|Dances with Wolves (1990)
```

(제가 작성한 쉘 스크립트의 7번 결과값)

```
Enter your choice [ 1-9 ] 7

Please enter the 'user id' (1~943):12

4|15|28|50|69|71|82|88|96|97|98|127|132|133|143|157|159|161|168|170|172|174|191|195|196|200|202|203|2
04|215|216|228|238|242|276|282|300|318|328|381|392|402|416|471|480|591|684|708|735|753|754

4|Get Shorty (1995)
15|Mr. Holland's Opus (1995)
28|Apollo 13 (1995)
50|Star Wars (1977)
69|Forrest Gump (1994)
71|Lion King, The (1994)
82|Jurassic Park (1993)
88|Sleepless in Seattle (1993)
96|Terminator 2: Judgment Day (1991)
97|Dances with Wolves (1990)
```

사용자가 8번을 선택함 -> 8번 : 20~29세이며 직업이 '프로그래머'인 사용자가 평가한 영화의 평균 평점을 계산하여 출력합니다.

```
    elif [ $choice = 8 ]; then
        echo
        read -p "Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'occupation' as 'programmer'?(y/n):" intention
        if [ $intention = "y" ]; then
            echo
            find_user_id=$(awk -F'|' '$2 >= 20 && $2 <= 29 && $4 == "programmer" {print $1;}' $user | sort -n)
            touch condensed_data_1.txt
            touch condensed_data_2.txt
            touch condensed_data_3.txt
            for i in ${find_user_id}; do
                awk -v user_id="$i" -F' ' '($1 == user_id) {print int($2),int($3)}' "$data" >> condensed_data_1.txt
            done
            awk -F' ' '{key=$1; sum[key]+=$2; count[key]++;} END {for(key in sum) {print key, sum[key], count[key];}}' condensed_data_1.txt >> condensed_data_2.txt
            awk -F' ' '{key=$1; if($2 > max[key]){max[key]=$2; line[key]=$0;}} END {for(key in max){print line[key];}}' condensed_data_2.txt |sort -n > condensed_data_3.txt
            awk -F' ' '{
    average = sprintf("%.5f",$2 / $3);
    sub(/\.?0+$/, "", average);
    printf "%d %s\n", $1, average;}' condensed_data_3.txt
            rm condensed_data_1.txt
            rm condensed_data_2.txt
            rm condensed_data_3.txt
        else
            continue
        fi
```

<8번 알고리즘>

1. 변수 find_user_id에 u.user 파일에서 20~29세이며 직업이 '프로그래머'인 사용자 ID를 찾아서 저장합니다.
2. 추출한 데이터를 축약해서 저장할 텍스트 파일 3개를 만듭니다.
3. for문을 돌려 변수 find_user_id에 안에 있는 사용자 ID들과 u.data파일 안에 있는 사용자 ID를 매칭시켜 20~29세이며 직업이 '프로그래머'인 사용자들이 본 영화 ID와 영화 평점들을 추출하여 condensed_data_1.txt에 저장합니다. 그러면 condensed_data_1.txt의 1번째 열의 정보는 영화 ID, 2번째열의 정보는 각 유저마다 각 영화를 매긴 평점들이 저장됩니다.
4. condensed_data_1.txt의 $1(영화 ID)가 같으면 평점들을 합하고 같은 ID가 매칭 될 때마다 count가 1씩 올라가도록 만들어 반복문을 통해 그것들을 condensed_data_2.txt에 저장합니다. 그러면 condensed_data_2.txt의 1째 열의 정보는 영화 ID, 2번째 열의 정보는 평점들의 총합, 3번째 열의 정보는 count로 저장이 됩니다. 이때, 영화 ID, 평점 총합, count를 계산하면서 영화 ID 중복값이 많습니다.
5. condensed_data_2.txt 중복을 없애기 위해 condensed_data_2.txt안에서 $2(영화 평점 총합)이 제일 크면 평점 총합과 count 계산이 다 끝낸 결과물이기 때문에 $2(영화 평점 총합)이 제일 큰 행만 남기고 다 지워서 중복하는 영화 ID를 다 삭제합니다. 그리고 그것들을 다 정렬해서 condensed_data_3.txt에 저장합니다. 그러면 condensed_data_3.txt에는 최종적으로 전부 오름차순으로 중복 없이 1번째 열의 정보는 영화 ID, 2번째 열의 정보는 영화의 평점 총합, 3번째 열의 정보는 영화를 본 유저 수 이런 식으로 데이터가 안에 있습니다.
6. 이 condensed_data_3.txt안에 있는 $2 / $3를 바로 연산해주어 과제 형식대로 각 영화마다 유저들이 매긴 평균 평점이 4초 내로 나오게 됩니다.

<8번 코드 설명>

-> touch : .txt 파일 생성합니다.

-> key=$1은 현재 레코드의 첫 번째 필드($1) 값을 key 변수에 할당합니다. 이것은 이후에 그룹화와 집계에 사용됩니다.

-> sum[key]+=$2는 sum 배열에서 key를 키로 사용하여 해당 필드($2) 값을 누적합니다. 이 부분은 특정 key에 대한 값을 누적하는 역할을 합니다.

-> count[key]++는 count 배열에서 key를 키로 사용하여 해당 키에 대한 레코드 개수를 카운트합니다. 이 부분은 그룹 내의 레코드 수를 추적합니다.

-> >> .txt 형태 : 스크립트의 출력 결과를 .txt 파일에 추가합니다. >>는 출력을 파일에 추가하는 리다이렉션 연산자입니다.

-> 생성한 txt파일은 rm 명령어로 제거해줍니다.

-> sub(/\.?0+$/, "", average) : sub(패턴, 대체할 것, 대체할 대상)

-> sub의 패턴 부분은 대체할 대상을 찾기 위한 정규 표현식 패턴을 나타냅니다.

-> sub의 대체 부분은 찾은 패턴을 어떤 식으로 대체할 것인지를 나타냅니다.

-> sub의 대상 부분은 패턴을 찾을 대상 문자열을 나타냅니다.

-> /\.?0+$/는 정규 표현식입니다.

-> \.? : 점(.) 문자가 0회 또는 1회 등장하는 것을 나타냅니다. \.?는 소수점(.) 문자가 없거나 한 번 나타나는 것을 의미합니다.

-> 0+ : 숫자 0이 1회 이상 반복되는 것을 나타냅니다. 즉, 하나 이상의 연속된 0을 의미합니다.

-> $: 패턴이 문자열의 끝에 매칭되어야 함을 나타냅니다. 이는 패턴이 문자열의 끝에 나타나는 것을 찾겠다는 것을 의미합니다.

(과제 8번 pdf)

```
Enter your choice [ 1-9 ] 8

Do you want to get the average 'rating' of
movies rated by users with 'age' between 20 and
29 and 'occupation' as 'programmer'?(y/n):y

1 4.29412
2 3
3 3.5
4 3.7
5 3.25
7 4.22222
8 3.5
9 4.1
10 4
11 4.3125
.
.
.
1512 3
1513 2
1518 4
1531 3
1552 2
1597 1
1600 4
1621 1
1655 2
```

(제가 작성한 쉘 스크립트의 8번 결과값)

```
Enter your choice [ 1-9 ] 8

Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 and 'oc
cupation' as 'programmer'?(y/n):y

1 4.29412
2 3
3 3.5
4 3.7
5 3.25
7 4.22222
8 3.5
9 4.1
10 4
11 4.3125
12 4.69231
13 3.375
14 4
15 3.85714
16 3
17 3.5
19 4
20 1
21 2.8
22 4.14286
23 4.57143
24 3.76923
25 3.38462
26 3.25
27 3
28 4
29 3.5
30 4
31 3.55556
32 5
33 4.2
34 4
35 1
```

```
1305 3
1311 3
1312 3
1314 2
1336 2
1376 1
1406 3
1407 1
1408 1
1410 3
1411 3.5
1412 1
1415 4
1419 1.33333
1425 2
1428 3
1437 2
1438 4
1439 4
1440 2.5
1443 5
1446 3
1456 4
1478 3
1480 1
1483 5
1485 3
1487 2
1491 1
1509 1
1512 3
1513 2
1518 4
1531 3
1552 2
1597 1
1600 4
1621 1
1655 2

Enter your choice [ 1-9 ]
```

사용자가 9번을 선택함 -> 9번 : 스크립트를 종료합니다.

```
    elif [ $choice = 9 ]; then
        echo "Bye!"
        exit
    else
        continue
    fi
done
```

<9번 알고리즘>

1. exit를 이용하여 현재 실행 중인 루프가 즉시 종료되며 스크립트도 종료하게 합니다.

(과제 9번 pdf)

```
--------------------------
User Name: fos
Student Number: 00000000
[ MENU ]
1. Get the data of the movie identified by a
specific 'movie id' from 'u.item'
2. Get the data of action genre movies from
'u.item'
3. Get the average 'rating' of the movie
identified by specific 'movie id' from 'u.data'
4. Delete the 'IMDb URL' from 'u.item
5. Get the data about users from 'u.user'
6. Modify the format of 'release date' in 'u.item'
7. Get the data of movies rated by a specific
'user id' from 'u.data'
8. Get the average 'rating' of movies rated by
users with 'age' between 20 and 29 and
'occupation' as 'programmer'
9. Exit
--------------------------
Enter your choice [ 1-9 ] 1

Please enter 'movie id'(1~1682):1

1|Toy Story (1995)|01-Jan-
1995||http://us.imdb.com/M/title-
exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0
|0|0|0|0|0|0

Enter your choice [ 1-9 ] 3

Please enter the 'movie id'(1~1682):1

average rating of 1: 3.87832

Enter your choice [ 1-9 ] 9
Bye!
```

(제가 작성한 쉘 스크립트의 9번 결과값)