OSS project 1

12191905 이인설

- 프로그램 설명

사용자로부터 입력을 받아 데이터로부터 사용자가 원하는 결과를 출력해주는 프로그램

- 사용자 입력에 대한 각 기능

사용자 입력	기능
1	원하는 영화의 아이디를 입력하면 해당 영화의 정보를 출력
2	액션 장르의 영화들 중 영화 아이디를 오름차순 기준 상위 10개에 대한 영화 아이디와 제목을 출력
3	원하는 영화 아이디를 입력하면 해당 영화의 평균 평점을 출력
4	영화 정보들 중 'IMDb URL' 정보를 제외하고, 영화 아이디의 오름차순 기준 상위 10개에 대한 영화 정보를 출력
5	유저 아이디의 오름차순 기준 상위 10개에 대한 유저 정보를 출력
6	영화 정보 중 'release date'의 형식을 YearMonthDate 형식으로 바꾸어 영화 아이디의 오름차순 기준 하위 10개에 대한 영화 정보 출력
7	원하는 유저의 아이디를 입력하면 해당 유저가 평점을 남긴 영화 아이디를 출력. 이후 영화 아이디의 오름차순 기준 상위 10개의 영화 제목을 출력
8	20대 프로그래머가 평점을 남긴 영화들의 평균 평점을 모두 출력
9	프로그램 종료

- 코드 세부 사항 설명

0. 전체적인 구조

```
#! /bin/bash

if [ $# -ne 3 ] ; then
        echo "usage: $0 u.item u.data u.user"
        exit 1
fi

# Print MENU

# Required Functions

# Required Vairables
start_movie_id=1
end_movie_id=1682
start_user_id=1
end_user_id=943
stop="N"

until [ "$stop" == "Y" ]
do
```

```
        read -p "Enter your choice [ 1-9 ] " n
        case $n in
        1)
        2)
        3)
        4)
        5)
        6)
        7)
        8)
        9)
                echo "Bye!"
                stop="Y"
                ;;
        *)
                echo "Please enter the number between 1 and 9"
                ;;
        esac
done
```

0-1) 제일 첫 줄에 Shabang을 통해 스크립트 파일을 실행할 interpreter를 지정해줍니다.

0-2) 이후 if문을 이용하여 shell script를 실행할 때 필요한 파일들을 제대로 명시하지 않으면 사용법을 출력해 제대로 실행할 수 있게 경고문을 출력하고 종료시키는 코드입니다.

0-3) 프로그램이 시작하면 메뉴창을 출력해주는 코드입니다.

0-4) 그 다음은 각 명령에 필요한 함수들을 구현한 코드들입니다.
(이후 2번에서 각 case에서 사용한 함수들을 같이 설명)

0-5) 그 다음은 각 명령에 필요한 변수들을 작성한 코드들입니다.

start_movie_id=1	영화 아이디의 가장 작은 수입니다.
end_movie_id=1682	영화 아이디의 가장 큰 수입니다.
start_user_id=1	유저 아이디의 가장 작은 수입니다.
end_user_id=943	유저 아이디의 가장 큰 수입니다.
stop="N"	until loop을 제어하는 변수입니다.

0-6) until loop을 사용해 사용자로부터 명령을 입력받을 수 있게 하였습니다.
until loop이 실행되면 사용자에게 명령을 받고 case 구문을 사용하여 각 입력(1~9)에 따른 명령을 수행하도록 만들었습니다.
case 1 ~ case 8은 각 기능에 맞게 구현하였습니다. (이후 2번에서 각 case를 설명)
case 9는 “Bye!”를 출력하고 until loop 제어 변수인 stop을 “Y”로 만들어 until loop을 종료시켜 프로그램을 종료합니다.
case *는 사용자가 1~9가 아닌 다른 입력을 하면 올바른 입력을 하라고 경고문을 출력하는 코드입니다.

1. 공통 함수

```
new_line() {
        echo -e "\n"
}

check_number() {
        local result
        while true
        do
                read -p "$1" result
                new_line
                if [ "$result" -ge "$2" ] && [ "$result" -le "$3" ]
                then break
                else echo -e "Please enter the number between $2 and $3\n"
                fi
        done
        return "$result"
}

check_answer() {
        local answer
        while true
        do
                read -p "$1" answer
                new_line
                if [ "$answer" == "y" ]; then return 0
                elif [ "$answer" == "n" ]; then return 1
                else echo -e "Please enter 'y' or 'n' only \n"
                fi
        done
}
```

1-1) new_line() (매개변수 없음)

출력 시 한 줄을 띄워주는 역할을 합니다.

1-2) check_number() (매개변수 : information, start_number, end_number)

사용자에게 영화 아이디 혹은 유저 아이디를 입력 받는 경우, 해당 입력이 올바른 범위에 있는지 확인하는 함수입니다.
인자로 입력 요구사항 문구, 아이디 시작 숫자, 아이디 끝 숫자를 줍니다.
인자로 받은 입력 요구사항 문구를 출력하며 사용자에게 입력을 받습니다.
해당 입력이 올바르지 않다면 시작 숫자와 끝 숫자 사이로 다시 입력하라고 요구합니다.
올바른 입력이 들어온다면 종료상태 값을 사용자의 입력으로 바꾸어 각 case들이 이를 사용할 수 있도록 하였습니다.

1-3) check_answer() (매개변수 : information)

사용자에게 선택한 명령을 수행할 것인지 묻는 경우, 올바른 입력이 들어왔는지 확인하는 함수입니다. 인자로 입력 요구사항 문구를 줍니다.
인자로 받은 입력 요구사항 문구를 출력하며 사용자에게 입력을 받습니다.
올바르지 않은 입력이 들어오면 y 또는 n으로 입력하라고 요구합니다.

y가 입력되면 종료 상태값을 0으로, n이 입력되면 종료 상태값을 1로 만듭니다.
이후 각 case에서 종료 상태값을 이용해 명령을 수행할지 판단합니다.
if문의 조건으로 해당 함수를 사용하면 해당 함수의 종료 상태값이 조건이 됩니다.

2. 각 case에 대한 설명

각 case에서 $1, $2, $3은 u.item, u.data, u.user에 해당됩니다.

- case 1 : 원하는 영화의 아이디를 입력하면 해당 영화의 정보를 출력

```
1)
        information="Please enter 'movie id' (1~1682) : "
        check_number "$information" $start_movie_id $end_movie_id
        movie_id=$?
        awk -F\| -v id=$movie_id '$1==id {print $0}' "$1"
        new_line
        ;;
```

case 1에 알맞은 입력 요구사항 문구와 영화 아이디의 처음과 끝 숫자를 check_number()의 인자로 주어 사용자에게 입력을 받습니다.
이후 check_number()의 종료 상태값을 이용해 입력받은 아이디를 movie_id에 저장합니다.
이후 awk를 이용해 u.item에서 입력받은 영화 아이디의 정보를 출력합니다.

- case 2 : 액션 장르의 영화들 중 영화 아이디를 오름차순 기준 상위 10개에 대한 영화 아이디와 제목을 출력

```
2)
        information="Do you want to get the data of 'action' genre movies from 'u.item'?(y/n) : "
        if check_answer "$information“
        then
                awk -F\| '$7==1 {print $1, $2}' "$1" | head -n 10
                new_line
        fi
        ;;
```

case 2에 알맞은 입력 요구사항 문구를 check_answer()의 인자로 주어, 해당 명령을 수행할지 if문의 조건으로 사용합니다.
명령을 수행하게 되면, awk를 이용해 u.item의 7번째 열인 액션 장르가 1인지 확인한 뒤, 첫 번째, 두 번째 열인 영화 아이디와 제목을 출력합니다.
이후 head -n을 이용해 영화 아이디의 오름차순 기준 상위 10개에 대한 항목만 출력합니다.

- case 3 : 원하는 영화 아이디를 입력하면 해당 영화의 평균 평점을 출력

```
3)
        information="Please enter the 'movie id' (1~1682) : "
        check_number "$information" $start_movie_id $end_movie_id
        movie_id=$?
        awk -v id=$movie_id '$2==id { sum+=$3; count+=1 }
         END { if (count>0)
                 {printf("average rating of %d : %.5f\n", id, sum/count)} }' "$2" |
         sed -E 's/\.?0*$//'
        new_line
        ;;
```

case 3에 알맞은 입력 요구사항 문구와 영화 아이디의 시작과 끝 숫자를 check_number()의 인자로 주어 사용자에게 입력을 받습니다.
이후 check_number()의 종료 상태값을 이용해 입력받은 아이디를 movie_id에 저장합니다.
이후 awk를 이용해 u.data의 데이터 중 2번째 열에 해당하는 movie id에서 사용자가 원하는 영화 아이디와 일치하는 데이터들을 찾습니다.
END를 이용해 찾은 데이터들의 3번째 열인 평점을 모두 더해 sum 변수에 저장하고 평점 개수를 count에 저장합니다.
이후 연산이 끝나면 printf를 이용해 소숫점 5번째자리까지의 평균 평점을 구합니다.
평균 평점을 구한 뒤, sed를 통해 의미 없는 0을 지워줍니다.
정규표현식의 의미는 다음과 같습니다.

정규표현식	설명
\.?	ex) 3.00000과 3.10000에 상관없이 의미없는 0을 지워야 하기 때문에 . 이 0번 또는 한번 들어갔는지 확인합니다.
0*$	ex) 3.00001의 경우 0은 의미없는 숫자가 아니며 3.00000의 경우 0은 의미없는 숫자입니다. 따라서 해당 문자열의 가장 끝인 소숫점 5번째자리까지 0이 계속 반복되는 경우만 의미없는 0입니다. 그렇기에 $를 이용해 의미없는 0을 처리하였습니다.

- case 4 : 영화 정보들 중 'IMDb URL' 정보를 제외하고, 영화 아이디의 오름차순 기준 상위 10개에 대한 영화 정보를 출력

```
4)
        information="Do you want to delete the 'IMDb URL' from 'u.item'?(y/n) : "
        if check_answer "$information"
        then
                sed -E 's/(http)+([^\|]*)\|/|/g' < "$1" | head -n 10
                new_line
        fi
        ;;
```

case 4에 알맞은 입력 요구사항 문구를 check_answer()의 인자로 주어, 해당 명령을 수행할지 if문의 조건으로 사용합니다.
명령을 수행하게 되면, standard input redirection을 이용해 u.item을 sed에게 넘겨줍니다.
이후 정규 표현식을 이용해 'IMDb URL' 정보를 추출합니다.

정규표현식	설명
(http)+	URL이 http로 시작하여 http를 그룹화하여 한개 이상 있는 부분을 그룹화
([^|]*)	u.itm은 | 를 기준으로 열이 구분되어 있기 때문에 |를 제외한 0개 이상의 문자열을 그룹화
|	'IMDb URL'의 열을 구분지어주는 마지막 | 를 의미

정규표현식에서 'IMDb URL'의 열을 구분 짓는 마지막 |를 포함했기 때문에 해당 부분을 모두 | 로 바꾸어 'IMDb URL' 부분을 제외해줍니다.
이후 head -n을 이용해 영화 아이디의 오름차순 기준 상위 10개에 대한 영화 정보를 출력합니다.

- case 5 : 유저 아이디의 오름차순 기준 상위 10개에 대한 유저 정보를 출력
case 5에서는 두 함수를 사용합니다.

```
change_sex_format() {
        sed -e 's/M/male/g' -e 's/F/female/g' < "$1"
}

change_user_format() {
        sed -E 's/([0-9]+)\|([0-9]+)\|([a-z]+)\|([a-z]+)\|([0-9]+)/user \1 is \2 years old \3 \4/g'
}
```

1) change_sex_format() (매개변수 : u.user 파일)
해당 함수는 u.user를 인자로 받아 M은 male로, F는 female로 sed를 이용해 바꾸어줍니다. u.user 파일에서 문자열이 있는 열이 gender와 occupation 밖에 없고, occupation은 모두 소문자들로 이루어져 있어 정규 표현식을 사용하지 않았습니다.
case 5에서 u.user의 gender의 형식을 바꾸는 하나의 command 역할로 사용됩니다.

2) change_user_format() (매개변수 없음)
해당 함수는 u.user의 데이터 형식에 맞게 sed를 이용해 원하는 형식에 맞게 출력할 수 있도록 하는 함수입니다.
sed의 정규 표현식에서 ([0-9]+)는 숫자로 구성된 열, ([a-z]+)는 문자로 구성된 열에 해당하며 각각 \|를 이용해 열을 구분하고 있습니다. 이후 각 그룹을 원하는 형식에 맞게 배치해줍니다.
case 5에서는 gender의 표현 형식이 바뀐 u.user의 데이터 형식을 바꾸는 하나의 command 역할로 사용됩니다.

실제 case 5에 해당하는 코드입니다.

```
5)
        information="Do you want to get the data about users from 'u.user'?(y/n) : "
        if check_answer "$information"
        then
                change_sex_format "$3" | change_user_format | head -n 10
                new_line
        fi
        ;;
```

case 5에 알맞은 입력 요구사항 문구를 check_answer()의 인자로 주어, 해당 명령을 수행할

지 if문의 조건으로 사용합니다.
해당 명령을 수행하면 change_sex_format으로 gender의 형식을 바꿔준 뒤, 파이프라인을 통하여 gender의 형식이 바뀐 u.user의 데이터 형식을 change_user_format으로 원하는 형식에 맞게 바꿔줍니다.
이후 head -n을 통해 유저 아이디의 오름차순 기준 상위 10개를 출력합니다.

- case 6 : 영화 정보 중 'release date'의 형식을 YearMonthDate 형식으로 바꾸어
영화 아이디의 오름차순 기준 하위 10개에 대한 영화 정보 출력
case 6에서는 하나의 함수를 사용합니다.

```
change_month() {
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Jan-)(.+)/\1\2\3-01-\5/g' < "$1" |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Feb-)(.+)/\1\2\3-02-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Mar-)(.+)/\1\2\3-03-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Apr-)(.+)/\1\2\3-04-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-May-)(.+)/\1\2\3-05-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Jun-)(.+)/\1\2\3-06-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Jul-)(.+)/\1\2\3-07-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Aug-)(.+)/\1\2\3-08-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Sep-)(.+)/\1\2\3-09-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Oct-)(.+)/\1\2\3-10-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Nov-)(.+)/\1\2\3-11-\5/g' |
        sed -Ee 's/(.*\|)(.*\|)(.*)(-Dec-)(.+)/\1\2\3-12-\5/g'
}
```

1) change_month() (매개변수 : u.item 파일)
해당 함수는 u.item을 인자로 받아 release date의 열에 해당하는 데이터 형식을 바꿔주는 역할을 합니다.
첫 sed는 standard input redirection을 이용해 인자로 받은 u.item을 이용하고, 파이프라인을 이용해 다음 sed에게 출력값을 넘겨주어 12월까지 모두 적용합니다.
release date의 열 전에 있는 movie title에서도 문자열이 나오며 영화 제목에 월 약자가 포함되어 있을 수 있기 때문에 정규 표현식을 사용합니다.

정규표현식	설명
(.*|)	u.item은 | 로 구분되어 있어, 어떠한 문자 이후 |로 끝나는 부분들을 그룹화하여 movie id와 movie title의 열을 그룹화
(.*)	release date의 date 부분을 그룹화
(-Month-)	release date의 -월 약자- 부분을 그룹화
(.+)	release date의 year부분과 그 이후 열들을 그룹화

case 6에서는 월 약자를 숫자로 바꿔주는 하나의 command 역할로 사용됩니다.

실제 case 6에 해당하는 코드입니다.

```
6)
        information="Do you want to Modify the format of 'release data' in 'u.item'?(y/n) : "
        if check_answer "$information"
        then
                change_month "$1" |
                  sed -Ee 's/(.*)([0-9]{2})\-([0-9]{2})\-([0-9]{4})(.*)/\1\4\3\2\5/g' | tail -n 10
                new_line
        fi
        ;;
```

```
```

case 6에 알맞은 입력 요구사항 문구를 check_answer()의 인자로 주어, 해당 명령을 수행할지 if문의 조건으로 사용합니다.
이후 change_month()를 이용해 u.item의 release date 열의 월 약자를 숫자로 바꿉니다.
파이프라인을 통해 바꾼 데이터를 sed를 이용해 년월일 형식으로 바꿔줍니다.
(.*)는 release date 열 앞의 부분을, date와 month는 숫자가 두 번 나오기 때문에 ([0-9]{2})로 그룹화하고 year는 숫자가 4번 나오기 때문에 ([0-9]{4})로 그룹화하여 -로 이어진 부분을 찾아줍니다. 이후 release date 열의 뒷 부분은 (.*)로 그룹화해줍니다. 이후 그룹을 이용해 원하는 형식으로 바꾼 뒤, tail -n을 이용해 영화 아이디의 오름차순 기준 하위 10개에 대한 영화 정보를 출력합니다.

- case 7 : 원하는 유저의 아이디를 입력하면 해당 유저가 평점을 남긴 영화 아이디를 출력. 이후 영화 아이디의 오름차순 기준 상위 10개의 영화 제목을 출력

case 7에서는 하나의 함수를 사용합니다.

```
find_movie_ids() {
        awk -v user_id="$2" '$1==user_id {printf("%d\n", $2)}' "$1" | sort -n
}
```

1) find_movie_ids() (매개변수 : u.data, user_id)
인자로 받은 u.data의 첫번째 열인 user id에서 인자로 받은 user_id에 해당하는 movie id들을 출력합니다. 이때 '\n'을 단위로 출력하고 sort -n을 이용해 추출한 영화 아이디들을 오름차순으로 정렬합니다.

실제 case 7의 코드입니다.

```
7)
        information="Please enter the 'user id'(1~942) : "
        check_number "$information" $start_user_id $end_user_id
        user_id=$?
        want_movie_ids=$(find_movie_ids "$2" "$user_id")

        echo "$want_movie_ids" | tr '\n' '|' | sed -E 's/\|$//g'
        new_line

        for i in $(seq 1 10)
        do
                want_id=$(echo "$want_movie_ids" | awk -v i="$i" 'NR==i {print $1}')
                awk -F\| -v want_id="$want_id" '$1==want_id {printf("%d|%s\n", $1, $2)}' "$1"
        done
        new_line
        ;;
```

case 7에 알맞은 입력 요구사항 문구와 유저 아이디의 처음과 끝 숫자를 check_number()의 인자로 주어 사용자에게 입력을 받습니다.
이후 check_number()의 종료 상태값을 이용해 입력받은 아이디를 user_id에 저장합니다.
이후 find_movie_ids()를 이용해 해당 유저가 평점을 남긴 영화 아이디들을 want_movie_ids에 저장합니다.
이후 want_movie_ids를 tr을 이용해 find_movie_ids()에서 '\n'으로 구분했던 것을 '|'으로

대체합니다. 이후 sed를 이용해 제일 뒤에 있는 '|'를 지워 원하는 형식으로 유저가 평점을 남긴 영화 아이디들을 나열합니다.
이후 for loop을 이용해 10번 반복하여 want_movie_ids에서 차례대로 영화 아이디를 가져와 want_id에 저장합니다.
이후 awk를 이용해 u.item의 첫번째 열인 movie id에서 want_id와 일치하는 행을 찾아 두 번 째 열인 movie title과 함께 출력합니다.

- case 8 : 20대 프로그래머가 평점을 남긴 영화들의 평균 평점을 모두 출력
case 8에서는 2개의 함수를 사용합니다

```
get_20sProgrammer_ids() {
        awk -F\| '$4=="programmer" && $2>=20 && $2<=29 {printf("%d ", $1)}' "$1"
}

get_and_makefile_movie_id_rated_by_20sProgrammer() {
        local movie_ids=""
        local file=""
        for user_id in $2
        do
                user_movie_id=$(awk -v id="$user_id" '$1==id {print $2}' "$1" )
                movie_ids=$(echo "$movie_ids"; echo "$user_movie_id")

                add_file=$(awk -v id="$user_id" '$1==id {print $2, $3}' "$1")
                file=$(echo "$file"; echo "$add_file")
        done

        echo "$file" > result

        movie_ids=$(echo "$movie_ids" | sort -n | uniq)
        echo "$movie_ids"
}
```

1) get_20sProgrammer_ids() (매개변수 : u.user)
해당 함수는 20대 프로그래머들의 유저 아이디를 추출하는 함수입니다. u.user를 인자로 받아 awk를 이용해 20대 프로그래머들의 유저 아이디를 스페이스를 기준으로 출력해줍니다.
case 8에서 20대 프로그래머들의 유저 아이디를 출력해주는 하나의 command 역할을 해줍니다.

2) get_and_makefile_movie_id_rated_by_20sProgrammer()
(매개변수 : u.data, 20s_programmer_ids)
해당 함수는 u.data와 20대 프로그래머들의 유저 아이드를 인자로 받습니다.
먼저 movie_ids와 file이라는 local 변수를 만들어 줍니다.
for loop을 이용해 u.data에서 20대 프로그래머들의 유저 아이디와 일치하는 행들의 2, 3번째 열인 movie id와 rating에 대한 정보를 add_file에, movie_id만 가지는 정보는 user_movie_id에 저장해 줍니다. 이 변수들을 file과 movie_ids 변수에 추가하면서 for loop를 수행합니다.
for loop이 끝나면 20대 프로그래머들이 평점을 남긴 영화 아이디와 평점을 result라는 파일에 저장하고, 영화 아이디를 중복 제거하여 출력해줍니다.
이때 result에 같은 영화 아이디가 있더라도 각기 다른 20대 프로그래머들에 의해 평가된 점

수들이 저장되어 있습니다.
case 8에서 하나의 command 역할을 하여 20대 프로그래머들이 평점을 남긴 영화의 아이디를 출력해주고, 평점 점수와 함께 파일에 저장해줍니다.

실제 case 8의 코드입니다.

```
8)
        information="Do you want to get the average 'rating' of movies rated by users with 'age' between 20 and 29 'occupation' as 'programmer'?(y/n) : "
        if check_answer "$information"
        then
                programmer_ids=$(get_20sProgrammer_ids "$3")
                movie_id_rated_by_20sProgrammer
                        =$(get_and_makefile_movie_id_rated_by_20sProgrammer
                            "$2" "$programmer_ids" )
                # 해당 라인은 보고서에서 가독성을 위해 라인을 분리한 것입니다.

                for movie_id in $movie_id_rated_by_20sProgrammer
                do
                        awk -v id="$movie_id" '$1==id { sum+=$2; count+=1 }
                         END { if (count>0) printf("%d : %.5f\n", id, sum/count)}' result |
                         sed -E 's/\.?0*$//'
                done
                rm result
        fi
        new_line
        ;;
```

case 8에 알맞은 입력 요구사항 문구를 check_answer()의 인자로 주어, 해당 명령을 수행할지 if문의 조건으로 사용합니다.
명령을 수행하게 되면 먼저 get_20sProgrammer_ids()를 통해 20대 프로그래머들의 유저 아이디를 추출합니다.
그 다음 get_and_makefile_movie_id_rated_by_20sProgrammer()를 통해 20대 프로그래머들이 평점을 남긴 영화 아이디를 추출하고 각 영화 아이디와 평점을 result 파일에 저장합니다. (사용자가 해당 프로그램을 사용할 때, 의도치 않게 해당 프로그램과 같은 디렉터리에 result라는 이름의 텍스트 파일이 있음에 대비해 확장자명을 주지 않았습니다.)
이후 for loop을 통해 result에서 각 영화 아이디에 해당하는 평점들의 평균을 구합니다.
평균을 구하는 코드는 case 3의 설명과 동일합니다.
평균을 모두 구했으면 더이상 result 파일이 필요하지 않아 삭제해줍니다.