OSS과제(Shell Script) README

1. **전체 구성**

```
#! /bin/bash
main() {
        echo "--------------------------"
        echo "User Name: JeongJunHan"
        echo "Student Number: 12200529"
        echo "[ MENU ]"
        echo "1. Get the data of the movie identified by a specific 'movie id' from 'u.item'"
          .
          .
        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 "--------------------------"

    while :
    do
        read -p "Enter your choice [ 1-9 ] " choice
        case $choice in
            1)
                echo
                read -p "Please enter 'movie id'(1~1682):" movie_id
                echo
                awk -F\| -v movie_id=$movie_id '$1 == movie_id {print}' $1
                echo
                ;;

              .
              .
              .

            9)
                echo "Bye!"
                exit 0
                ;;
            *)  # Invalid choice
                echo "Invalid choice"
                ;;
        esac
    done
}

main $1 $2 $3
```

구성: main함수, `main $1 $2 $3` 커맨드(파일 3개를 인자로 받음)

설명: main 함수는 처음에 메뉴가 출력되고, 그 후에는 while문을 돌면서 choice를 계속 입력받고 그에 따른 case문을 실행한다.

2. **1번 기능**

```
1)
    echo
    read -p "Please enter 'movie id'(1~1682):" movie_id
    echo
    awk -F\| -v movie_id=$movie_id '$1 == movie_id {print}' $1
    echo
    ;;
```

새로 알게 된 기능

1) `cat echo $1 | awk …` 대신에 awk가 읽어올 수 있게 끝에 $1을 붙여주면 된다.

`awk -F\| -v movie_id=$movie_id '$1 == movie_id {print}' $1`

2) -v로 변수 사용 가능

3) -F\| 로 '|' 구분기호 사용 가능

설명: u.item의 첫번째 열과 사용자가 입력한 movie_id와 같으면 그 행 전체를 출력함

3. **2번 기능**

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

설명: u.item의 7번째 열(action)이 1인 것들만 1,2번째 행을 출력 | 정렬 | 위의 10개만 출력

4. **3번 기능**

```
3)
    echo
    read -p "Please enter the 'movie id'(1~1682):" movie_id
    echo
    avg_rating=$(awk -F'\t' -v id=$movie_id '$2 == id { sum += $3; count++ } END { if (count > 0) print sum/count }' $2)
    echo "average rating of $movie_id: $avg_rating"
    echo
    ;;
```

설명: avg_rating은 u.data의 2번째 열과 사용자 입력이 같으면 sum, count에 적절한 값을 더해주고, count가 0보다 크면 소수점 5째자리까지 반올림되서 표현되게 출력

새로 알게된 점: awk 안에서 계산하면 자동으로 반올림되어 5째 자리까지 출력된다.

5. **4번 기능**

```
4)
    echo
    read -p "Do you want to delete the 'IMDb URL' from 'u.item'?(y/n):" confirm
    echo
    if [ "$confirm" = "y" ]
    then
        sed 's/^\(\([^|]*|\)\{4\}\)[^|]*|/\1|/' $1 | head -10
        echo
    fi
    ;;
```

설명: `[^|]*|` ('|'가 아닌 문자)* 형식에 |로 끝나는 문자를 4번 반복하고 `[^|]*|` 가 나오는 큰 문자열을 \1(4번 반복했던 것)|로 대체해서 5번째 열(url)의 값을 삭제하였다. 뒤에 head -10 처리했다.

6. **5번 기능**

```
5)
    echo
    read -p "Do you want to get the data about users from 'u.user'?(y/n):" confirm
    echo
    if [ "$confirm" = "y" ]
    then
        sed -n 's/^\([0-9]*\)|\([0-9]*\)|\([MF]\)|\([^|]*\)|.*/user \1 is \2 years old \3 \4/p' $3 | sed 's/ M/ male/' | sed 's/ F/ fe
        echo
    fi
    ;;
```

설명: u.user의 user id | age | gender | occupation | zip code에서 `user \1 is \2 years old \3 \4` 특정한 문자열로 바꿨고 M,F를 male,female로 바꿨다.

7. **6번 기능**

```
6)
    echo
    read -p "Do you want to Modify the format of 'release data' in 'u.item'?(y/n):" confirm
    echo
    if [ "$confirm" = "y" ]
    then
        sed -n 's|\([0-9][0-9]\)-\(...\)-\([0-9][0-9][0-9][0-9]\)|\3\2\1|p' $1 | tain -n 10 | sed 's/Jan/01/;s/Feb/02/;s/Mar/03/;s/Apr/
        echo
    fi
    ;;
```

새로운 기능: s/from/to/ 형식 대신 s|from|to| 사용해도 상관없음

설명: u.item의 release date 부분의 01-Jan-1995를 19950101 같은 형식으로 바꾸는 코드이다.
숫자 2글자, 문자 3글자, 숫자4글자 양식의 순서를 바꿔주고 후에 각 문자에 따라 숫자로 대체해주었다. 마지막 10라인을 출력했다.

8. **7번 기능**

```
7)
    echo
    read -p "Please enter the ‘user id’(1~943):" user_id
    echo
    movie_ids=$(awk -v user="$user_id" '$1 == user {print $2}' $2 | sort -n)
    echo "$movie_ids" | tr '\n' '|' | sed 's/|$/\n/'
    echo
    top_10_ids=$(echo "$movie_ids" | head -10)
    awk -F\| -v top10="$top_10_ids" 'BEGIN { split(top10, arr, "\n"); for(i in arr) lookup[arr[i]] } $1 in lookup { print $1"|"$2 }' $1
    echo
    ;;
```

새로운 기능: END 처럼 BEGIN도 앞에서 전처리 1번을 할 수 있다.

설명

1) movie_ids에 u.data 파일의 첫번째 열과 사용자 입력이 같으면 2번째 열(movie id)를 출력하는 값을 저장한다.
`tr '\n' '|'` 로 엔터를 |로 바꿔주고 `sed 's/|$/\n/'` 로 마지막 |를 줄바꿈으로 바꿔서 출력해준다.

2) movie_ids의 상위 10개만 가져온다.
arr배열에 top10을 “\n”을 기준으로 잘라서 저장하고, arr의 모든 i(인덱스)에 대해 top10개의 값을 lookup의 배열의 키로 저장한다.
lookup에 u.item의 1번째 열의 값이 존재하면 u.item의 1,2번째 열을 출력한다.

9. **8번 기능**

```
8)
    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 'prograr
    echo
    if [ "$confirm" = "y" ]; then
        user_ids=$(awk -F\| '$2 >= 20 && $2 <= 29 && $4 == "programmer" {print $1}' $3)
        awk -F'\t' -v users="$user_ids" 'BEGIN { split(users, arr, "\n"); for (i in arr) userLookup[arr[i]] } $1 in userLookup { sum[$2
        END {
            for (movie in sum) {
                avg = sum[movie]/count[movie];
                print movie, avg
            }
        }' $2 | sort -k1,1n
        echo
    fi
    ;;
```

설명

1) user_ids에 u.user에서 20대이면서 programmer인 id들을 저장한다.

2) Begin에서 id들을 arr배열에 저장하고 userLookup 배열에 키의 존재유무를 저장한다.
u.data에서 1번째 열(user id)가 userLookup의 키로 존재하면, 키에 대한 sum에 rating을 더해주고, count수를 늘려준다.

3) End에서 sum의 키(movie id)를 movie에 for문으로 할당하고 각각의 movie에 대한 평균을 구해준다. print하면 자동으로 5째 자리까지 반올림되어서 출력된다.

4) 후에 1번째 열을 기준으로 k1,1 정렬을 해주고 n은 키를 숫자로 인식하게 하는 것이다. `-k1,1n`

10. **9번 기능과 추가 처리**

```
9)
    echo "Bye!"
    exit 0
    ;;
*)  # Invalid choice
    echo "Invalid choice"
    ;;
```

설명: 9번 기능은 Bye 출력후 종료하고 추가 기능으로 Invalid Choice를 입력받았을 때, “Invalid choice”를 출력하고 다시 while문에 의해 choice를 사용자로부터 입력받는다.