Project 1Repotrt

Problem 1

For printing the detailed information by movie id, we receive input from user

```
read -p "Please enter 'movie id'(1~1682):" id
```

According to `u.item`, the id field is the first one, so we use

```
# turn on -v option to specify a variable and -F option to sepecify the delimiter
awk -v ID=${id} -F'|' '$1==ID {print$0}' $1
```

to print the whole information.

Problem 2

we can see that action genre is in the 7_{th} field delimited by `|`, so we check whether the 7_{th} is `1` or not, if so we print the first field (the movie id) and the second field (the movie name).

```
# $7 represents whether the movie is action genre while $1 and $2 represent id and name, respectively.
awk -F'|' '$7==1 {print $1,$2}' $1 | head -n 10
```

Since we are only interested in the first ten lines of the result, we can use pipe and `head` to extract the first ten lines.

Problem 3

`u.data` contains user id, movie id and scores rated by the user, which are delimited by space. We are interested in the given movie id, so we should extract the corresponding movie id which is in the second field.

Since `awk` manipulates data line by line, we declare two variables, `sum` and `n`, to remain the total scores and how many user has rated the movie.

```
# Note: %g helps us format the result automatically
awk -v ID=${id} '$2==1 {sum+=$3; n+=1} END {printf("average rating of %d: %g\n", ID, sum / n)}' $2
```

The `END` block will be executed only when `awk` has extracted all data.

Problem 4

We use pipe to transfer data flow and `sed` to substitute data. We want to delete URL, it is equivalent to substitute all URL with nothing. In the single-quoted part, `s` represents substitution, `http:[^\|]*//g` is the string which we want to substitute. This is, the string begins with `http:` and ends with `|`. The regular expression `[^\|]*` helps us to match the string. The argument `g` represents global substitution.

```
# http:[^\|]* matches the string which begins with http: and ends with '|'
# Note: we have to escape '|', since '|' is logic or.
cat $1 | sed -E 's/http:[^\|]*//g' | head -n 10
```

Problem 5

First, we use `sed -E 's/\|/ /g'` to remove all delimiters such that we can manipulate data easily. Right now, the fields are delimited by space, so we can use back-references to permute the ordering or to be used for formatting the resulting string. Each one associated with a pair of parentheses corresponds to `\1`, `\2`, `\3` and `\4`, respectively. Since we are only interested in the first four fields, we can use `.+` to ignore the remaining string.

```
cat $3 | sed -E 's/\|/ /g' |
sed -E 's/(\w+) (\w+) (\w+) (\w+).+/user \1 is \2 years old \3 \4/' |
head -n 10 |
sed -E 's/F/female/g' |
sed -E 's/M/male/g'
```

After extracting the first ten lines of the resulting strings, we use `sed` to replace `F` and `M`, respectively, with `female` and `male`.

Problem 6

The data has the form of `<day>-<month>-<year>` which is delimited by `-`. `<day>` consists of only 2 numbers, so we can use `[0-9]{2}` to locate the day. Similarly, using `\w{3}` and `[0-9]{4}` locates month and year. Then, use back-reference to permute the ordering. In the end, we use a series of pipe commands to substitute moth with the corresponding number.

```
cat $1 | sed -E 's/([0-9]{2})-(\w{3})-([0-9]{4})/\3\2\1/' | \
tail -n 10 | \
sed -E 's/Jan/01/' | \
sed -E 's/Feb/02/' | \
sed -E 's/Mar/03/' | \
sed -E 's/Apr/04/' | \
sed -E 's/May/05/' | \
sed -E 's/Jun/06/' | \
sed -E 's/Jul/07/' | \
sed -E 's/Aug/08/' | \
sed -E 's/Sep/09/' | \
sed -E 's/Oct/10/' | \
sed -E 's/Nov/11/' | \
sed -E 's/Dec/12/'
```

Problem 7

First, we extract all ids of the movies rated by the given user. Then, sort the exacted ids and use `tr` to replace all newline character `\n` with `|`. Since there is a `|` at the end of the resulting output, we use `sed =E 's/\|$/\n'` to substitute the end `|` with newline character.

```
awk -v UID=${uid} '$1==UID {print $2}' $2 | sort -n | tr '\n' '|' | sed -E 's/\|$/\n/'
```

For convenience of manipulating data, we redirect the output to a file.

```
file="rated_id.txt"
awk -v UID=${uid} '$1==UID {print $2}' $2 | sort -n >${file}
```

We specify two files as the arguments of `awk`, they will be processed one by one. We check which file is being processed by using built-in variables `NR` and `FNR`. `NR` represents how many lines `awk` has processed, while `FNR` represents which line of the file is being processed. Thus, we can compare `NR` with `FNR` to check which file is being processed. For buffering data, we use built-in hash map, which maps field 1 to field 2 (mapping id to its name).

```
awk -F'|' 'NR==FNR {a[$1]=$2} NR>FNR {printf("%d|%s\n", $0, a[$0])}' $1 ${file} | head -n 10
```

When processing the file which stores the sorted id, we can directly map the id to its name.

In the end, we delete the temporary file.

```
if [ -f ${file} ]; then
  rm ${file}
fi
```

Problem 8

Similarly, we use a temporary file to buffer data. First, we extract all ids of the users, whose age is between $20 \sim 29$ and whose occupation is programmer, to the buffer file. Before calculating the results, we initialize some variables. `sum[i]` represents the sum of scores of the movie of `id i`. `cnt[i]` represents how many people has rated the movie. `status[i]` represents whether the user whose id is `i` satisfied the condition. Since the first argument of `awk` is `${file}` which stores the ids of the users who satisfy the condition, we mark the user by assigning 1 to `status[i]`. Then, we traverse `u.data`, if the user id is marked, then we sum the scores of the corresponding movie and increment the counter.

```
file="programmer_id.txt"
awk -F'|' '$2>=20 && $2<=29 &&$ 4~"programmer" {print $1}' $3 > ${file}
awk 'BEGIN{
    for (i = 1; i < 1682; ++i)
    {
        sum[i] = 0;
        cnt[i] = 0;
    }
}
NR==FNR {status[$1]=1} NR>FNR && status[$1]==1 {sum[$2] += $3; cnt[$2]++;}
END {
    for (i = 1; i < 1682; ++i)
        if (sum[i] != 0) printf("%d %g\n", i, sum[i] / cnt[i]);
}' ${file} $2
```

After processing the two files, we use our buffering data to calculate the average rating.