**REPORT**

상징, 로고, 엠블럼, 등록 상표이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| **과목명** | 오픈소스SW개론 |
| **학과** | 컴퓨터공학과 |
| **학번** | 12184876 |
| **이름** | 오영수 |
| **제출일자** | 2024.05. |

* Except Handling

if [ "$#" -ne 3 ]; then

echo "usage: $0 file1 file2 file3"

exit 1

fi

1. “$#”를 통해 인자의 개수가 3개 아니라면 문제의 조건에 따라 ‘usage: $0 file1 file2 file3’를 출력하고 프로그램을 종료하는 코드를 짰습니다.

* Start my script

echo "\*\*\*\*\*\*\*\*\*\*\*\*OSS1 - Project1\*\*\*\*\*\*\*\*\*\*\*\*"

echo "\* StudentID : 12184876 \*"

echo "\* Name : 오영수 \*"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

1. 위의 Exception Handling에서 인자 3개가 들어온다면 문제의 조건에 따라 해당 내용을 출력하게 끔 구현하였습니다.

* Print\_menu

function print\_menu() {

echo "[MENU]"

echo "1. Get the data of Heung-Min Son's Current Club, Appearances, Goals, Assists in players.csv"

echo "2. Get the team data to enter a league position in teams.csv"

echo "3. Get the Top-3 Attendance matches in matches.csv"

echo "4. Get the team's league position and team's top scorer in teams.csv & players.csv"

echo "5. Get the modified format of date\_GMT in matches.csv"

echo "6. Get the data of the winning team by the largest difference on home stadium in teams.csv & matches.csv"

echo "7. Exit"

}

* 메인 실행 루프

while true; do

print\_menu

echo -n "Enter your CHOICE (1~7) : "

read choice

case $choice in

1) son\_data ;;

2) team\_data ;;

3) top\_attendance ;;

4) team\_and\_scorer ;;

5) modify\_date ;;

6) largest\_win ;;

7)

echo "Bye!"

break

;;

\*) echo continue ;;

esac

done

1. While true를 통해 Exit기능을 수행할 수 있는 7번 case를 제외하고 무한루프를 돌 수 있게 설정하였습니다. (7번 case는 Bye!를 출력하고 break문을 통해 while문 탈출)
2. 위에서 나타낸 코드인 print\_menu 함수를 통해 각각 메뉴에 대한 설명을 출력하고, 이후 choice를 read해서 choice가 어떤 숫자인지에 따라서 case문을 통해 각각 문제에 맞는 함수를 구현하였습니다.
3. Project 설명 pdf에서 Repeat until 7(Exit)이라 되어 있어서 1~7 이외의 숫자는 다시 while문을 돌게 continue문으로 설정해주었습니다.

TEAMS\_FILE=$1

PLAYERS\_FILE=$2

MATCHES\_FILE=$3

1. ./2024-OSS-Project1.sh teams.csv players.csv matches.csv로 실행함에 따라 파일 이름 변수를 위와 같이 할당해주었습니다.

* 각각 함수들에 대한 설명(7번은 앞에서 설명했으므로 생략했습니다.)

1. son\_data()

function son\_data() {

echo -n "Do you want to get the Heung-Min Son's data? (y/n)"

read answer

if [[ $answer == "y" ]]; then

awk -F, '$1=="Heung-Min Son" {

print "Team:" $4 ", Appearance:" $6 ", Goal:" $7 ", Assist:" $8

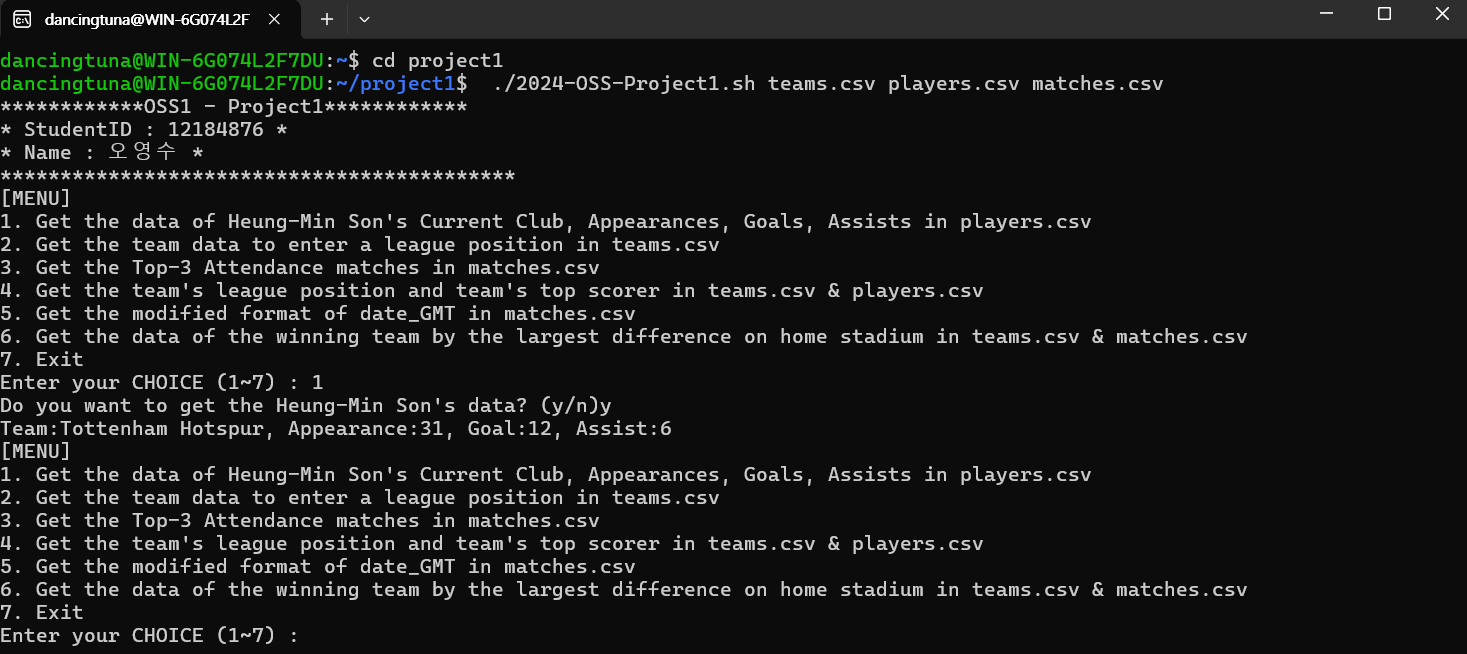
}' "$PLAYERS\_FILE"

fi

}

1. answer를 read해주고 y이면 if문을 수행을 합니다.
2. awk 명령어를 통해 구분자를 ‘.’로 설정해주고, PLAYERS\_FILE를 읽었을 때, 첫번째 field인 full\_name에 “Heung-Min Son"이 있다면 4번째 field인 Current Club과 6번째 field인 appearances\_overall, 7번째 field인 goals\_overall, 8번째 field인 assists\_overall를 문제에 주어진 조건에 따라 출력해주는 함수를 구현해주었습니다.

- 실행 결과



2. team\_data()

function team\_data() {

echo -n "What do you want to get the team data of league\_position[1~20]: "

read position

awk -F, -v pos="$position" '$6 == pos {

wins=$2; draws=$3; losses=$4

total\_matches=wins+draws+losses

if (total\_matches > 0) {

win\_rate=wins/total\_matches

} else {

win\_rate=0

}

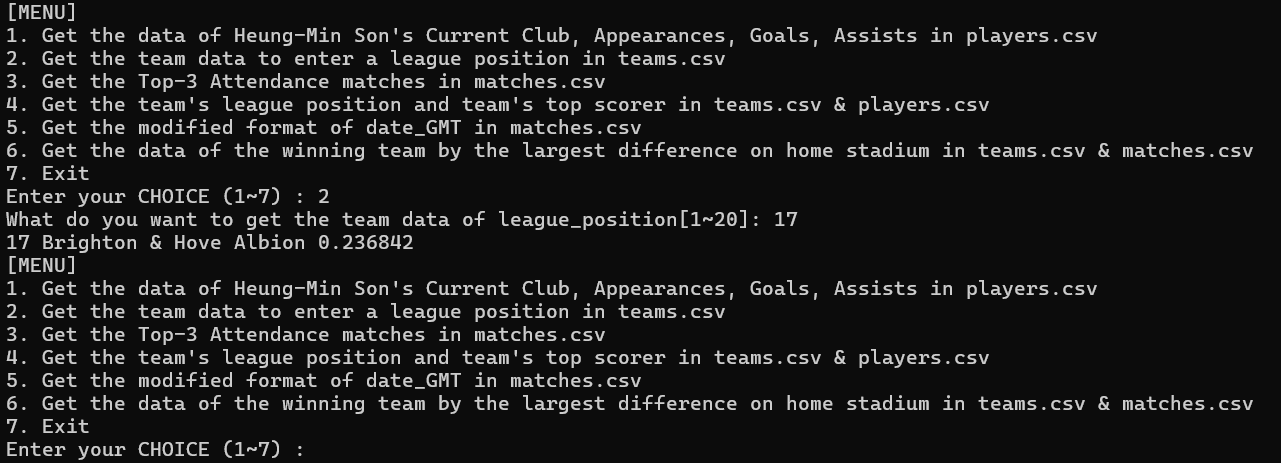
printf "%s %s %.6f\n", $6, $1, win\_rate

}' "$TEAMS\_FILE"

}

1. position을 read해주고, awk 명령어를 통해 구분자를 ‘,’로 설정하고, ‘-v pos=$position’을 통해 사용자 입력을 받을 걸 awk 스크립트 내에서 사용할 변수 ‘pos’에 전달하였습니다.
2. TEAMS\_FILE에서 league\_position은 6번째 field이기 때문에 $6 == pos를 통해 사용자가 입력한 position정보가 일치할 때, 2번째 field인 wins와, 3번째 field인 draws와 4번째 field인 losses를 각각 변수에 저장을 하였습니다.
3. win\_rate는 문제의 조건에 따라 wins/total\_matches로 구현을 해주었고, total match가 아예 없는 팀이 있다면 승률이 0%이기 때문에 else문을 통해 처리해주었습니다.
4. printf를 통해 결과를 출력해주는데, league\_position, common\_name, win\_rate를 출력해주는데, **이때, pdf에서 나온 것과 같이 소수점 여섯째 자리까지 출력을 하기 위해서 %.6f로 출력을 해주었습니다.**

- 실행 결과



3. top\_attendance

function top\_attendance() {

echo -n "Do you want to know Top-3 attendance data and average attendance? (y/n): "

read answer

if [[ $answer == "y" ]]; then

echo -e "\*\*\*Top-3 Attendance Match\*\*\*\n"

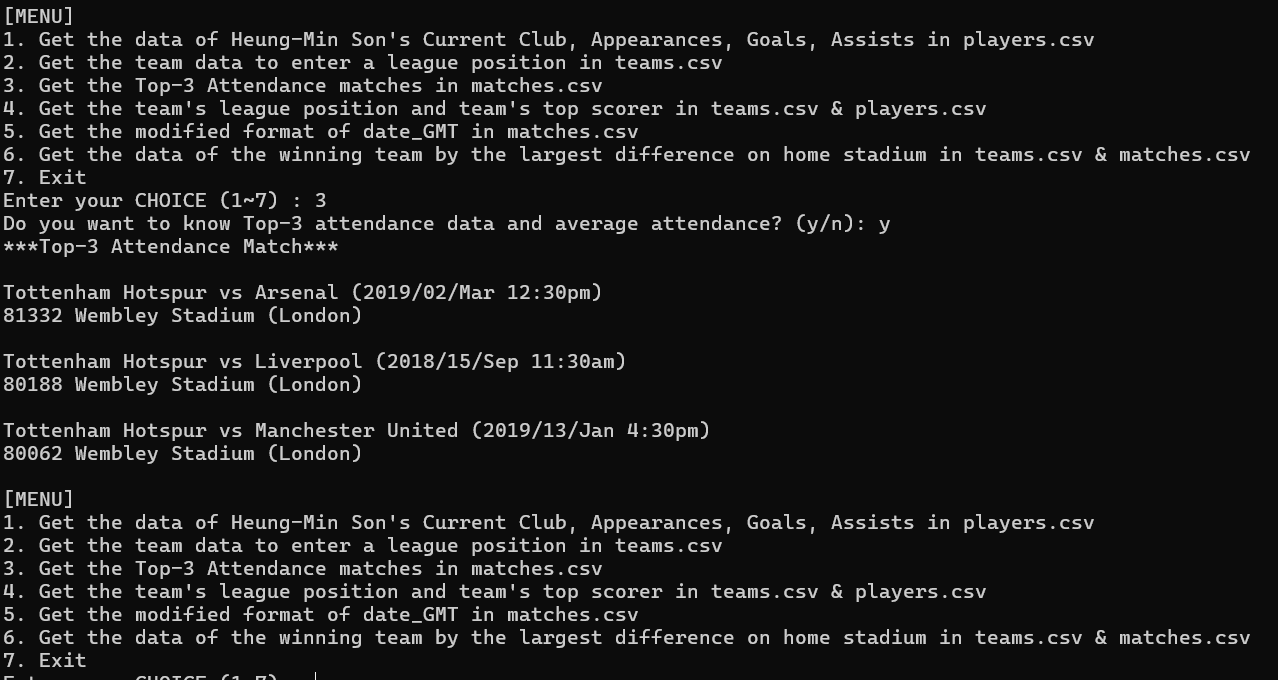
awk -F, 'NR > 1 {print $2, $0}' $MATCHES\_FILE | sort -nr -k1,1 | cut -d ' ' -f2- | head -3 |

awk -F, '{printf "%s vs %s (%s)\n%s %s\n\n", $3, $4, $1, $2, $7}'

fi

}

1. answer를 read해주고 y이면 if문을 수행을 합니다.
2. awk 명령어를 통해 구분자를 ‘,’로 설정해주고, 첫번째 행(헤더)을 제외하는 NR>1를 설정해주었습니다.
3. 첫번째 print에서 print $2, $0를 실행하면 첫번째 filed에 attendance, 2번째 field에 전체 field가 들어오게 됩니다.
   1. Ex) 74439 2018/10/Aug 7:00pm,74439,Manchester United,Leicester City,2,1,Old Trafford (Manchester)
4. 그리고 관중 수인 첫번째 filed를 ‘sort -nr -k1, 1’를 통해 정렬을 해주고, ‘cut -d ‘ ‘ -f2-‘를 통해 정렬된 결과에서 ‘ ‘ 빈칸을 구분자로 첫번째 field를 제외하고 나머지 부분을 추출한 다음 ‘head -3’를 통해 상위 3개의 결과만 선택을 해줍니다.
5. 이후 구분자 ‘,’를 통해 printf를 실행해 주면 home\_team\_name vs away\_team\_name (date\_GMT)  
    attendance stadium\_name 이 나오게 됩니다.

- 실행 결과

4. team\_and\_scorer

function team\_and\_scorer() {

echo -n "Do you want to get each team's ranking and the highest-scoring player? (y/n) : "

read answer

if [[ $answer == "y" ]]; then

echo -e "\n"

sort -t, -k6,6n $TEAMS\_FILE | while IFS=, read -r common\_name wins draws losses points\_per\_game league\_position cards\_total shots fouls; do

if [[ "$league\_position" != "league\_position" ]]; then

max\_goals=$(awk -F, -v team="$common\_name" '$4==team {print $7}' $PLAYERS\_FILE | sort -nr | head -1)

top\_scorers=$(awk -F, -v team="$common\_name" -v max\_goals="$max\_goals" '$4==team && $7==max\_goals {printf "%s %s ", $1, $7}' $PLAYERS\_FILE)

echo -e "$league\_position $common\_name\n$top\_scorers\n"

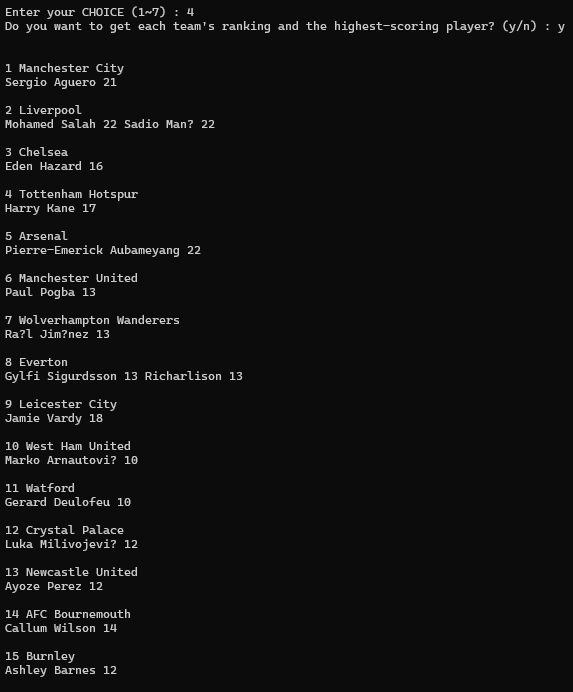
fi

done

fi

}

1. answer를 read해주고 y이면 if문을 수행을 합니다.
2. ‘sort -t, -k6,6n $TEAMS\_FILE’를 통해 TEAMS\_FILE의 필드 구분자 ‘,’를 기준으로 6번째 field인 league\_position를 기준으로 오름차순 정렬해주었습니다.
3. 이후 while IFS=, read -r를 통해 정렬된 파일을 읽으며 각각 선언된 변수에 맞게 할당해주었습니다.
4. if [[ "$league\_position" != "league\_position" ]]를 통해 TEAMS\_FILE의 첫번째 줄인 헤더를 제외시켜 주고, max\_goals, top\_scorers를 계산하여 문제의 주어진 조건에 맞게 출력하였습니다.
5. max\_goals는 awk 명령어를 통해 구분자를 ‘,’로 잡고, -v team="$common\_name"를 통해 common\_name의 값을 team에 할당하였고, 이후 '$4==team {print $7}'를 통해 소속된 팀의 이름인 $4가 team과 일치할 경우 7번째 field인 goals\_overall를 출력한 다음, sort -nr | head -1를 통해 -nr로 역순 정렬과 첫번째 줄을 반환하는 것을 통해 가장 많은 골을 넣은 플레이어의 골수를 max\_goal에 저장했습니다.
6. Top\_scorers는 awk 명령어를 통해 구분자를 ‘,’로 잡고, -v team="$common\_name"를 통해 common\_name의 값을 team에 할당, - max\_goals="$max\_goals"를 통해 위에서 찾은 data인 max\_goals 의 값을 max\_goals 에 할당했습니다.
7. '$4==team && $7==max\_goals를 통해 PLAYERS\_FILE에서 소속팀의 이름인 $4가 team과 일치하고, goals\_overall이 maxgoal이면 printf를 통해 선택된 선수의 이름 $1과 골수인 $7를 포맷팅해줬습니다. 이를 통해 top scorer가 여러 명이면 여러 명 출력할 수 있게 하였습니다.(Liverpool case)

- 실행 결과

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

5. modify\_date

function modify\_date() {

echo -n "Do you want to modify the format of date? (y/n) : "

read answer

if [[ $answer == "y" ]]; then

awk -F, 'NR > 1 && NR <= 11 { print $1 }' matches.csv |

awk -F'[/ ]' '{ print $1 "/" $2 "/" $3 " " $4 }' |

sed -e 's/Jan/01/' -e 's/Feb/02/' -e 's/Mar/03/' \

-e 's/Apr/04/' -e 's/May/05/' -e 's/Jun/06/' \

-e 's/Jul/07/' -e 's/Aug/08/' -e 's/Sep/09/' \

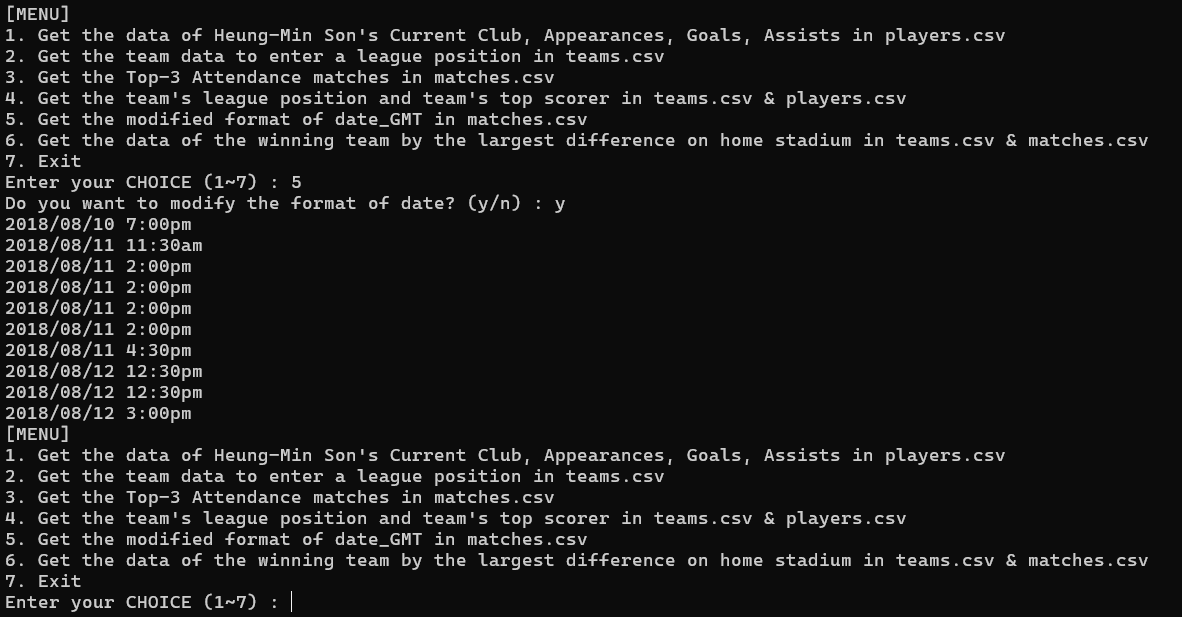
-e 's/Oct/10/' -e 's/Nov/11/' -e 's/Dec/12/' |

awk -F'[/ ]' '{ printf $1 "/" $3 "/" $2 " " $4 "\n" }'

fi

}

1. answer를 read해주고 y이면 if문을 수행을 합니다.
2. Awk 명령어로 ‘,’를 구분자로 설정하고, 문제에서 주어진 10개의 data를 추출하기 위해서 헤더를 제외한 2번째에서 11번째 행을 추출을 하고 첫번째 field인 date\_GMT를 포맷팅해주었습니다.
3. awk -F'[/ ]' '{ print $1 "/" $2 "/" $3 " " $4 }'를 통해 구분자를 ‘/’와 ‘ ‘(빈칸)으로 설정해주고, $1을 년도, $2을 일, $3을 월, $4를 시간으로 포맷팅해줍니다.
4. 이후 sed 명령어를 통해 월 이름을 숫자로 바꿔주는 작업을 진행했습니다.
5. awk -F'[/ ]' '{ printf $1 "/" $3 "/" $2 " " $4 "\n" }'를 통해 구분자를 ‘/’와 ‘ ‘(빈칸)으로 설정해주고, 문제에 주어진 조건과 같이 년/월/일 시간 형식으로 맞춰 출력하게 구현을 하였습니다.



6. largest\_win

# part 1

function largest\_win() {

teams=$(awk -F, 'NR > 1 {printf "%2d) %-35s\n", NR-1, $1}' $TEAMS\_FILE)

echo "$teams" | awk '{

lines[NR] = $0;

}

END {

half = int((NR + 1) / 2); # 배열의 절반 계산

for (i = 1; i <= half; i++) {

if (i + half <= NR) {

printf "%-40s %s\n", lines[i], lines[i + half]; # 첫 절반과 두 번째 절반 출력

} else {

printf "%-40s\n", lines[i]; # 남은 팀 출력 (홀수 경우)

}

}

}'

1. 해당 함수는 길이가 길어서 2개의 부분으로 나눠서 설명을 드리겠습니다.
2. 먼저 TEAMS\_FILE을 awk 명령어를 통해 각 팀의 이름인 $1를 출력하는데 NR-1를 통해 줄 번호를 1부터 시작하게 조정해주었습니다, 이때 %-35s를 통해 왼쪽 정렬을 하기 위해 구현했습니다.
3. 이후 이전에 생성한 teams 변수를 이용해 팀 이름을 문제에 주어진 조건에 맞춰 출력을 해주는 알고리즘을 구현했습니다.
4. 구현 방법은 ‘lines[NR]=$0;’을 통헤 각 라인을 ‘lines’ 배열에 저장해주고, half = int((NR + 1) / 2);를 해줍니다.
5. 이후 prinf를 통해 출력을 진행해주는데 각 라인마다 ‘lines[i], lines[i + half];’를 통해 각각 예를 들면 1과 11번째 팀 출력, 2와 12번째 팀 출력.. 이렇게 문제에 주어진 형식에 맞게 출력을 해줍니다.
6. 그 밑에 else문은 teams.csv file에서 팀의 개수가 홀수인 경우에 마지막에 하나만 남게 되므로 이를 고려하여 구현을 진행했습니다.

# part 2

echo -n "Enter your team number: "

read team\_number

team\_name=$(awk -F, -v num="$team\_number" 'NR == num + 1 {print $1}' $TEAMS\_FILE)

largest\_wins=$(awk -F, -v team="$team\_name" '

$3 == team {

goal\_diff = $5 - $6

print goal\_diff "," $0

}' $MATCHES\_FILE | sort -t, -nr -k1,1)

largest\_diff=$(echo "$largest\_wins" | head -1 | cut -d, -f1)

awk -F '[,/]' -v diff="$largest\_diff" '$1 == diff {

split($4, date\_parts, " ");

month = date\_parts[1];

time = date\_parts[2];

printf "\n%s %02d %s - %s\n", month, $3, $2, time;

printf "%s %d vs %d %s\n\n", $6, $8, $9, $7;

}' <<<"$largest\_wins"

1. 다음은 team의 숫자를 받아서 해당 팀이 홈으로 이긴 경기 중 가장 득실차가 큰 경기를 출력하는 메서드입니다.
2. 먼저 team\_number를 사용자 입력을 받고, 해당 숫자에 맞는 팀의 common\_name을 team\_name에 저장합니다.
3. 다음 largest\_wins는 우선 해당 team이 홈팀에서 이긴 경기들을 모두 찾고, 이후 sort를 통해 이를 득실차 기준으로 내림차순 정렬을 합니다.
4. Largest\_diff는 이를 바탕으로 head -1 | cut -d, -f1를 통해 가장 큰 득실차가 몇인지 추출을 해줍니다.
5. 이제 다음으로 awk 명령어를 통해 data를 추출하여 형식에 올바르게 출력이 되게끔 구현을 했는데, 현재 다음 largest\_wins의 정보는 대략 이런 형식입니다.

2,2018/01/Dec 3:00pm,25098,Crystal Palace,Burnley,2,0,Selhurst Park (London)

2,2019/02/Feb 3:00pm,25355,Crystal Palace,Fulham,2,0,Selhurst Park (London)

2,2019/30/Mar 3:00pm,25193,Crystal Palace,Huddersfield Town,2,0,Selhurst Park (London)

2,2019/12/May 2:00pm,25433,Crystal Palace,AFC Bournemouth,5,3,Selhurst Park (London)

1. 여기서 awk 명령어를 통해 구분자를 ‘,’와 ‘/’로 구분을 한다면, $1는 largest\_diff, $2는 년도, $3는 날짜, $4는 ‘month time’, $5는 attendance, $6는 home\_team\_name, $7는 away\_team\_name, $8는 home\_team\_goal\_count, $9는 away\_team\_goal\_count가 됩니다.
2. 이제 $4에서 ‘month time’을 split명령어를 통해 “ “ 빈 공백으로 나누어 dateparts라는 배열에 저장을 해준다면, dateparts[1]는 month부분, dateparts[2]는 time부분으로 나누어주었습니다.
3. 이제 각각 필요한 data가 다 추출되었으니 prinf 명령어를 통해 문제의 조건에 맞게 출력해주는 메서드를 구현했습니다.

- 실행 결과

