# Homework 2B

2021315385
이건 / Gun Daniel Lee

# Problem explanation

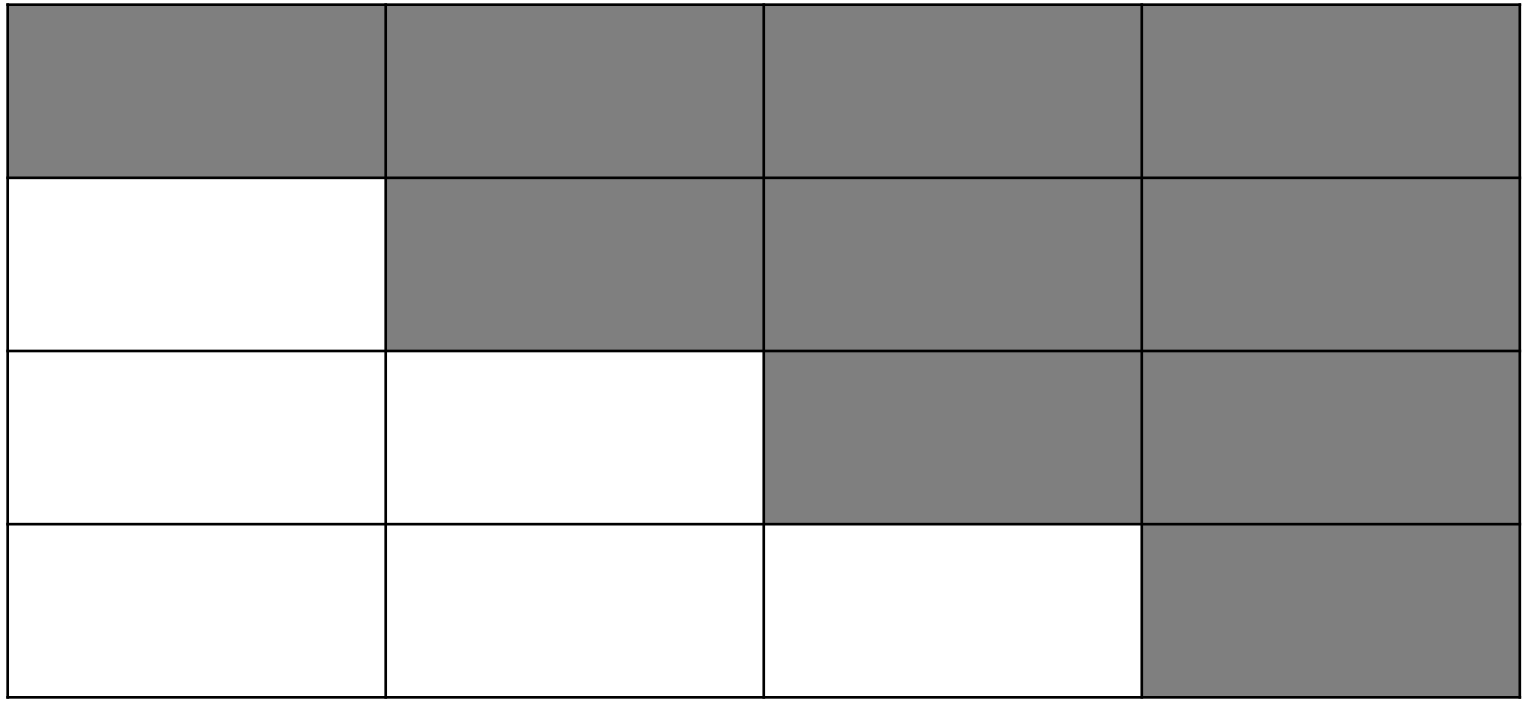
- Create all the game results that yield the maximum or minimum points (D) of a team with the Cth largest scores

- In other words, Maximize or minimize the Cth ranking's score

- Given inputs are:
  - A: points given for a Win
  - B: points given for a Tie
  - C: target ranking which needs to be maximized / minimized
  - D: 0 (minimum) or 1 (maximum)

# Solution explanation

- Focus on half of the matrix only (upper right division from the diagonal)
- Including the diagonal for the X's



# Solution explanation

- Maximum:
  - Have a tie score for Cth ranking and above (To replace all losses of the Cth ranking with ties)
  - Win 20 – C games and tie C – 1 games

- Minimum:
  - Have a tie score for Cth ranking and below (To eliminate any wins for C)
  - Solve for maximum for (C – 1) ranking
  - Lose C – 1 games and tie 20 – C games

# Solution explanation

- To simplify the code, both the maximum and minimum will apply

- For maximum, maximum code will apply to C, and minimum to C + 1

- For minimum, minimum code will apply to C, and maximum to C - 1

# Solution explanation

```c
#include <stdio.h>

int main()
{
  int a, b, c, d;
  char results[20][20];
  int num_of_ties;
  int i, j;

  scanf("%d %d %d %d", &a, &b, &c, &d);

  for(i = 0; i < 20; i++)
  {
    //num of ties for the rankings for > / >= c
    if(d == 1)
      num_of_ties = c;
    else
      num_of_ties = c - 1;

    for(j = i; j < 20; j++)
    {
      //diagonal is X
      if(i == j)
      {
        results[i][j] = 'X';
      }
      //have C - 1 number of ties
      else if(num_of_ties - 1 - i > 0)
      {
        results[i][j] = 'T';
        results[j][i] = 'T';
        num_of_ties--;
      }
      //for minimum, everything else is a tie
      else if(i >= num_of_ties)
      {
        results[i][j] = 'T';
        results[j][i] = 'T';
      }
      //remaining are wins
      else
      {
        results[i][j] = 'W';
        results[j][i] = 'L';
      }
    }
  }

  for(i = 0; i < 20; i++)
  {
    for(j = 0; j < 20; j++)
    {
      printf("%c", results[i][j]);
    }
    printf("\n");
  }
}
```

# Solution explanation

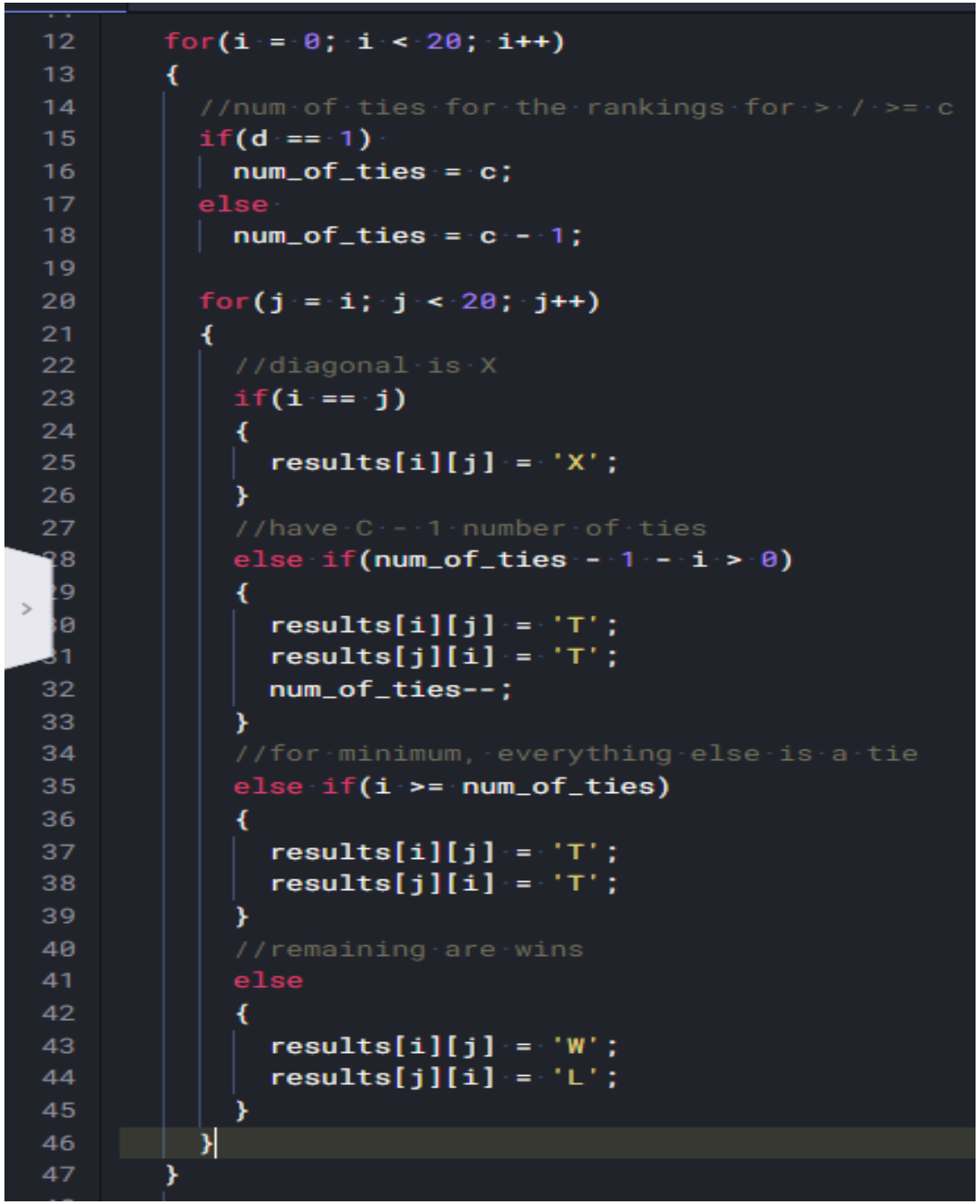
- A: points for Wins
- B: points for Losses
- C: target ranking
- D: 0 (minimum) or 1 (maximum)
- results: result matrix
- num_of_ties: number of ties of the winners
- i, j: iteration variables

```c
#include <stdio.h>

int main()
{
  int a, b, c, d;
  char results[20][20];
  int num_of_ties;
  int i, j;

  scanf("%d %d %d %d", &a, &b, &c, &d);

```

# Solution explanation

- For loop:

- Fill up the results matrix with W, L, T, X



```
for(i = 0; i < 20; i++)
{
  //num of ties for the rankings for > / >= c
  if(d == 1)
    num_of_ties = c;
  else
    num_of_ties = c - 1;

  for(j = i; j < 20; j++)
  {
    //diagonal is X
    if(i == j)
    {
      results[i][j] = 'X';
    }
    //have C - 1 number of ties
    else if(num_of_ties - 1 - i > 0)
    {
      results[i][j] = 'T';
      results[j][i] = 'T';
      num_of_ties--;
    }
    //for minimum, everything else is a tie
    else if(i >= num_of_ties)
    {
      results[i][j] = 'T';
      results[j][i] = 'T';
    }
    //remaining are wins
    else
    {
      results[i][j] = 'W';
      results[j][i] = 'L';
    }
  }
}
```

# Solution explanation

- ‘i’ For loop:
- If D = 1(max)
  - Apply max to c
- If D = 0(min):
  - Apply max to c - 1

```
12    for(i = 0; i < 20; i++)
13    {
14      //num of ties for the rankings for > / >= c
15      if(d == 1)
16        num_of_ties = c;
17      else
18        num_of_ties = c - 1;
19
```

# Solution explanation

- ‘j’ For loop:
- if case: diagonals = ‘X’
- 1st else if case: Ties for the rankings above c
- 2nd else if case: Ties for the rankings below c
- else case: W for the max cases, L for the min cases

```
for(j = i; j < 20; j++)
{
  //diagonal is X
  if(i == j)
  {
    results[i][j] = 'X';
  }
  //have C - 1 number of ties
  else if(num_of_ties - 1 - i > 0)
  {
    results[i][j] = 'T';
    results[j][i] = 'T';
    num_of_ties--;
  }
  //for minimum, everything else is a tie
  else if(i >= num_of_ties)
  {
    results[i][j] = 'T';
    results[j][i] = 'T';
  }
  //remaining are wins
  else
  {
    results[i][j] = 'W';
    results[j][i] = 'L';
  }
}
```

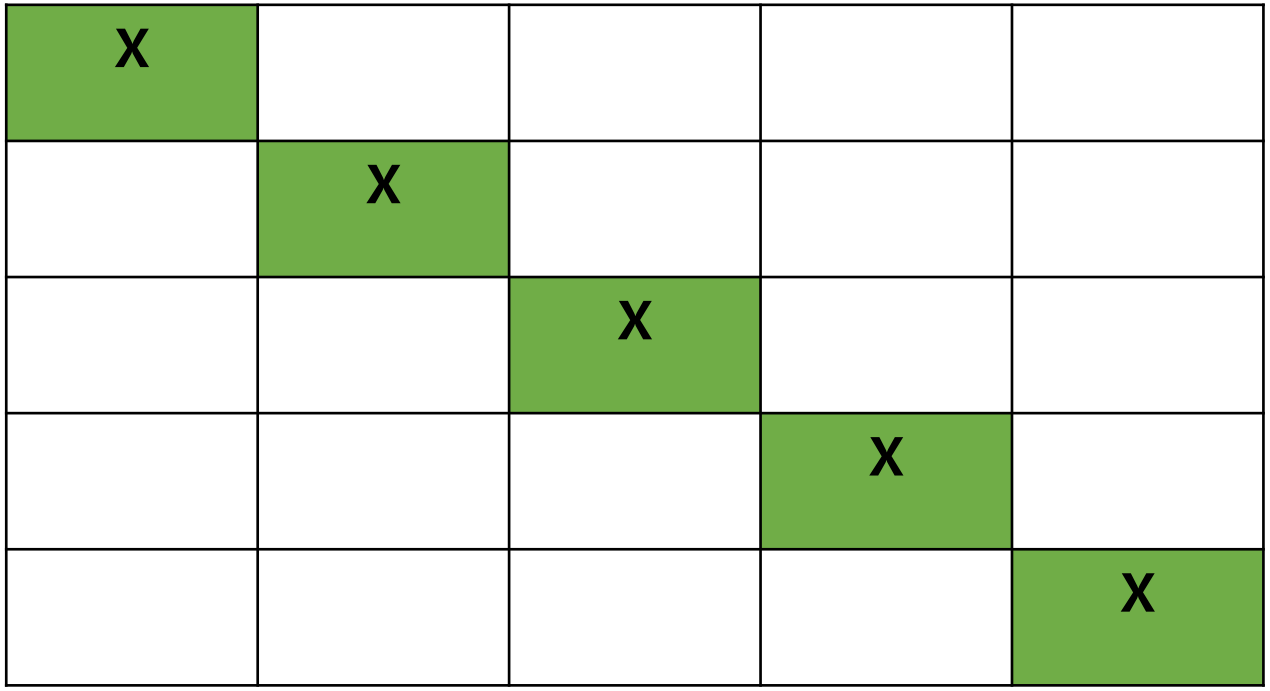
# Solution explanation

- Example for 5 by 5
- Input: 3 1 2 1

- For c = 2 and d = 1, number of ties needed = c – 1 = 1

# Solution explanation

- If cases: Fill diagonal with X



# Solution explanation

- 1st else if case: since max case, fill ties of ranks above c inclusive ( >= c)

| | | | | |
|---|---|---|---|---|
| X | T | | | |
| T | X | | | |
| | | X | | |
| | | | X | |
| | | | | X |

# Solution explanation

- 1st else if case: since max case, fill ties of ranks above c inclusive ( >= c)
- Since there can be no more ties, move on to the next

| | | | | |
|---|---|---|---|---|
| X | T | | | |
| T | X | | | |
| | | X | | |
| | | | X | |
| | | | | X |

# Solution explanation

- 2nd else if case: Fill up ties for all the ranks below c

| | | | | |
|---|---|---|---|---|
| X | T | | | |
| T | X | | | |
| | | X | T | T |
| | | T | X | T |
| | | T | T | X |

# Solution explanation

- Else case: Fill up all the blanks in the upper left half with ‘W’ and the lower half with L

| | | | | |
|---|---|---|---|---|
| X | T | W | W | W |
| T | X | W | W | W |
| L | L | X | T | T |
| L | L | T | X | T |
| L | L | T | T | X |

# Solution explanation

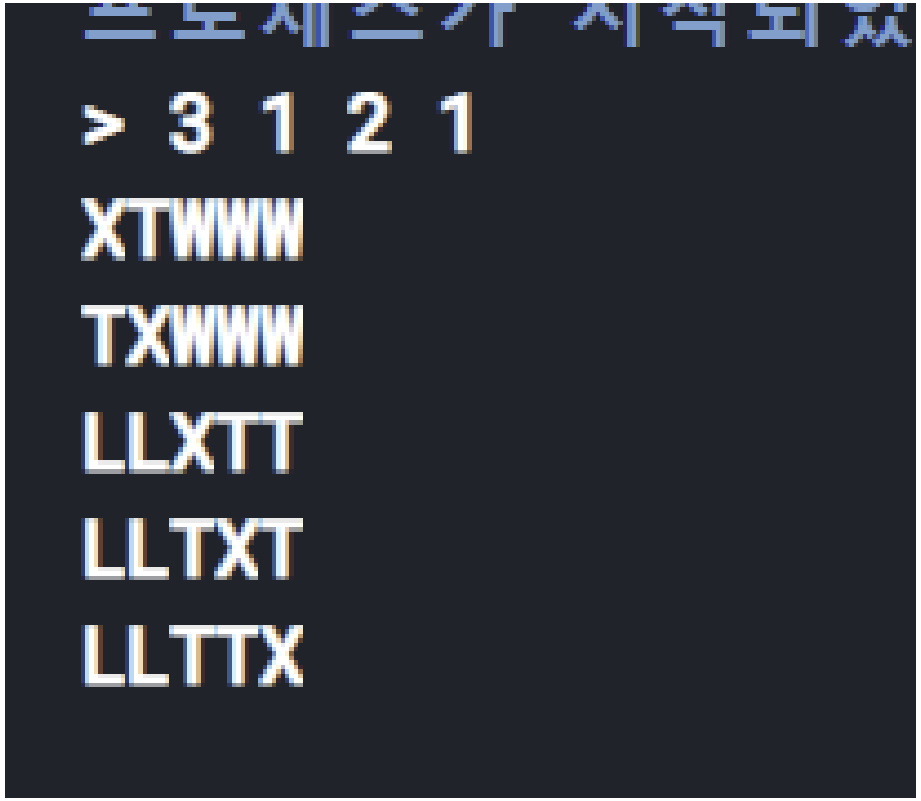
- Result Matrix

| | | | | |
|---|---|---|---|---|
| **X** | **T** | **W** | **W** | **W** |
| **T** | **X** | **W** | **W** | **W** |
| **L** | **L** | **X** | **T** | **T** |
| **L** | **L** | **T** | **X** | **T** |
| **L** | **L** | **T** | **T** | **X** |

# Solution explanation

- Result Matrix

| X | T | W | W | W |
|---|---|---|---|---|
| T | X | W | W | W |
| L | L | X | T | T |
| L | L | T | X | T |
| L | L | T | T | X |



# Solution analysis

- Main Idea was to simplify the code by using the nested for loop for half of the matrix only

- Time Complexity: O(n log(n))

# Solution analysis

- Pros: reduces complexity by only solving for half of the matrix to calculate the remaining half
  - Resulting matrix is easy to interpret due to a huge tie for both max and min
- Cons: No diversity in results
  - All of the results are technically set from the beginning

Thank you!