How to compile and run

1) gcc -o solver solver.c 10-to-0-by-1-or-2.c
2) ./solver 10

```
10: win
9: lose
8: win
7: win
6: lose
5: win
4: win
3: lose
2: win
1: win
0: lose
```

1) gcc -o solver solver.c 25-to-0-by-1-or-3-or-4.c
2) ./solver 25

```
25: lose
24: win
23: win
22: win
21: win
20: lose
19: win
18: win
17: win
16: win
15: lose
14: win
13: win
12: win
11: win
10: lose
9: win
8: win
7: win
6: win
5: lose
4: win
3: win
2: win
1: win
0: lose
```

solver.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// choose one to include from and comment out the other one
#include "10-to-0-by-1-or-2.h"
// #include "25-to-0-by-1-or-3-or-4.h"

int solve(int position)
{
    // win: 0, tie: 1, lose: 2
    int mem[position + 1];
    int prim;
    int status;
    int *moves;
    int temppos;

    for (int i = 0; i <= position; i++) // go from 0 to the position to dp the problem
    {
        prim = primitive_value(i);
        if (prim != 3) // check if prim val
        {
            mem[i] = prim;
        }
        else // not a prim val
        {
            moves = generate_moves(i); // generate possible moves from current posision, number of moves is stored in moves[0]
            if (moves[0] == -1)
            {
                free(moves);
                return -1;
            }

            status = 2;

            for (int j = 1; j <= moves[0]; j++) // collect all the new positions generated by the moves
            {
                temppos = do_move(i, moves[j]);

                if (mem[temppos] == 2) // at least one lose child == winning
                {
                    status = 0;
                    break;
                }
                else if (mem[temppos] == 1) // at least one tie child wout a lose child == tie
                {
                    status = 1;
                }
            }

            mem[i] = status;

            free(moves);
        }
    }

    return mem[position];
}

int main(int argc, char *argv[])
{
    if (argc > 1)
    {
        int max = atoi(argv[1]);
        int code;
        char *sol;
        for (int i = max; i >= 0; i--)
        {
            code = solve(i);
            if (code == -1)
            {
                sol = "error";
            }
            else if (code == 0)
            {
                sol = "win";
            }
            else if (code == 1)
            {
                sol = "tie";
            }
            else if (code == 2)
            {
                sol = "lose";
            }
            else
            {
                sol = "N/A";
            }
            printf("%d: %s\n", i, sol);
        }
    }
    else
    {
        printf("Please enter a max value!");
    }

    return 0;
}
```

10-to-0-by-1-or-2.c

```
#include <stdio.h>
#include <stdlib.h>

#include "10-to-0-by-1-or-2.h"

int do_move(int position, int move)
{
    // check all illegal moves
    if (!(0 < move && move < 3) || !(0 < position && position <= 10) || (position == 1 && move == 2))
    {
        return -1;
    }
    return position - move;
}

int *generate_moves(int position)
{
    // first elemnt of moves
    int *moves = malloc(3 * sizeof(int));
    if (!(0 < position && position <= 10))
    {
        moves[0] = -1;
    }
    else if (position == 1)
    {
        moves[0] = 1;
        moves[1] = 1;
    }
    else
    {
        moves[0] = 2;
        moves[1] = 1;
        moves[2] = 2;
    }

    return moves;
}

int primitive_value(int position)
{

    // win: 0, tie: 1, lose: 2, not_primitive: 3
    if (position == 0)
    {
        return 2;
    }
    else
    {
        return 3;
    }
}

// int main(int argc, char *argv[])
// {
//     printf("do_move(1, 2): %d\n", do_move(1, 2));

//     int *moves = generate_moves(3);
//     printf("Calling generate_moves(3): \n");
//     if (moves[0] == -1)
//     {
//         printf("no moves available, primitive values \n");
//     }
//     else
//     {
//         for (int i = 1; i <= moves[0]; i++)
//         {
//             printf("move %d: %d\n", i, moves[i]);
//         }
//     }
//     free(moves);

//     printf("primitive_value(0): %d\n", primitive_value(0));
//     printf("primitive_value(5): %d\n", primitive_value(5));

//     return 0;
// }
```

25-to-0-by-1-or-3-or-4.c

```c
#include <stdio.h>
#include <stdlib.h>

#include "25-to-0-by-1-or-3-or-4.h"

int do_move(int position, int move)
{
    // check all illegal moves
    if (!(0 < move && move < 5) || !(0 < position && position <= 25))
    {
        return -1;
    }
    return position - move;
}

int *generate_moves(int position)
{
    // first elemnt of moves
    int *moves = malloc(5 * sizeof(int));
    if (!(0 < position && position <= 25))
    {
        moves[0] = -1;
    }
    else if (position == 1)
    {
        moves[0] = 1;
        moves[1] = 1;
    }
    else if (position == 2)
    {
        moves[0] = 2;
        moves[1] = 1;
        moves[2] = 2;
    }
    else if (position == 3)
    {
        moves[0] = 3;
        moves[1] = 1;
        moves[2] = 2;
        moves[3] = 3;
    }
    else
    {
        moves[0] = 4;
        moves[1] = 1;
        moves[2] = 2;
        moves[3] = 3;
        moves[4] = 4;
    }

    return moves;
}

int primitive_value(int position)
{

    // win: 0, tie: 1, lose: 2, not_primitive: 3
    if (position == 0)
    {
        return 2;
    }
    else
    {
        return 3;
    }
}

// int main(int argc, char *argv[])
// {
//     printf("do_move(1, 2): %d\n", do_move(1, 2));

//     int *moves = generate_moves(3);
//     printf("Calling generate_moves(3): \n");
//     if (moves[0] == -1)
//     {
//         printf("no moves available, primitive values \n");
//     }
//     else
//     {
//         for (int i = 1; i <= moves[0]; i++)
//         {
//             printf("move %d: %d\n", i, moves[i]);
//         }
//     }
//     free(moves);

//     printf("primitive_value(0): %d\n", primitive_value(0));
//     printf("primitive_value(5): %d\n", primitive_value(5));

//     return 0;
// }
```

10-to-0-by-1-or-2.h

```
#ifndef _10_TO_0_BY_1_OR_2_H_
#define _10_TO_0_BY_1_OR_2_H_

#include <stdio.h>
#include <stdlib.h>

int do_move(int position, int move);

int *generate_moves(int position);

int primitive_value(int position);

#endif /* _10_TO_0_BY_1_OR_2_H_ */
```

25-to-0-by-1-or-3-or-4.h

```
#ifndef _25_TO_0_BY_1_OR_2_OR_3_OR_4_H_
#define _25_TO_0_BY_1_OR_2_OR_3_OR_4_H_

#include <stdio.h>
#include <stdlib.h>

int do_move(int position, int move);

int *generate_moves(int position);

int primitive_value(int position);

#endif /* _25_TO_0_BY_1_OR_2_OR_3_OR_4_H_ */
```