TPP2020-HW1-40847041S 朱自宇（請記得修改這裡！未填學號與姓名將會扣很大！若為多人作業，請列出所有成員。）

```
/* ------------------------------------------------------------------------------------------------------------------
// maze.h
// ----------------------------------------------------------------------------------------------------------------*/

/*
*	maze.h
*
*	made by 40847041S 朱自宇
*		A basic header file for 2D maps (or mazes.)
*
*	The maze can:
*	1. set a road symbol
*	2. set an obstacle symbol
*/

#ifndef MAZE_H
#define MAZE_H

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

class Maze
{
public:
	// setting the maze
	void setHeight( int k ) { height__ = k; }			// height
	void setWidth( int k ) { width__ = k; }			// width
	void setRoad( char k ) { road__ = k; }			// road
	void setObstacle( char k ) { obstacle__ = k; }		// obstacle
	void createMap();	// create space for the map
	void setMap();		// set up map, notice that you must call createMap() before


	// get values
	int getHeight() const { return height__; }
	int getWidth() const { return width__; }
	int getSize() const { return height__*width__; }
	char getRoad() const { return road__; }
	char getObstacle() const { return obstacle__; }
	const char *getMap() const { return map__; }
	void printMap() const;


	// checking
	bool isObstacle( int x, int y ) const { return ( map__[y*width__ + x] == obstacle__ ); }

	// after using, please remember to free the map !!!
	void freeMap() { free(map__); }

private:
	int height__;		// height of the maze
	int width__;		// width of the maze
	char road__;		// character that represent roads
	char obstacle__;	// character that represent obstacles
	char *map__;		// pointer to the maze
};
```

```
#endif

/* -----------------------------------------------------------------------------------------------------------------
// maze.cpp
// ---------------------------------------------------------------------------------------------------------------*/

#include <maze.h>

void Maze::createMap()
{
	map__ = (char *) calloc( height__ * width__, sizeof(char) );
}

void Maze::setMap()
{
	int count = 0;
	for( int i = 0; i < height__; i+=1 )
	{
		for( int j = 0; j < width__+1; j+=1 )
		{
			char c = fgetc(stdin);
			if( c == '\n' );
			else
				map__[i*width__+j] = c;
		}
	}
}

void Maze::printMap() const
{
	for( int i = 0; i < height__; i+=1 )
	{
		for( int j = 0; j < width__; j+=1 )
		{
			printf("%c", map__[i*width__+j]);
		}
		puts("");
	}
}

/* -----------------------------------------------------------------------------------------------------------------
// robot.h
// --------------------------------------------------------------------------------------------------------------- */

/*
*	robot.h
*
*	made by 40847041S  朱自宇
*		A very easy header design for robots who lives in a
*	2-dimensional world that can do some easy commands.
*
*	The robot can:
*	1. moves around by 4 directions (since it lives in a 2D world.)
*	2. Look at 4 directions: [ 0:North, 1:East, 2:South, 3:West ]
*/

#ifndef ROBOT_H
#define ROBOT_H
```

TPP2020-HW1-40847041S 朱自宇 （請記得修改這裡！未填學號與姓名將會扣很大！若為多人作業，請列出所有成員。）

```
#include <stdio.h>

class Robot
{
public:
        // basic settings about the robot
        void setposX( int x ) { posX__ = x; }
        void setposY( int y ) { posY__ = y; }
        void setLook( int f ) { look__ = f; }


        // get robot status
        int getposX() const { return posX__; }    // get the X-coordinate of the robot
        int getposY() const { return posY__; }    // get the Y-coordinate of the robot
        int getlook() const { return look__; }    // get the direction where the robot is facing [ 0:North,
1:East, 2:South, 3:West ]


        /* manipulating the robot */


        // human-like moving (consider facing direction)
        void GoForward( const int steps );
        void GoBackward( const int steps );
        void TurnRight() { look__ = (look__+1) % 4; }
        void TurnLeft() { look__ = (look__+3) % 4; }
        void TurnBack() { look__ = (look__+2) % 4; }

        // moving by coordinate, ignores facing direction
        void GoEast( int steps ) { posX__ += steps; }
        void GoWest( int steps ) { posX__ -= steps; }
        void GoNorth( int steps ) { posY__ += steps; }
        void GoSouth( int steps ) { posY__ -= steps; }

        // looking
        void LookAt( int direction ); // change looking direction of the robot [ 0:North, 1:East, 2:South,
3:West ]

private:
        // coordinate
        int posX__;             // x position of the robot
        int posY__;             // y position of the robot

        // the robot looks at one direction
        int look__ = 0;      // [ 0:North, 1:East, 2:South, 3:West ]
        // switch direction
        void LookNorth() { look__ = 0; }    // look North (up)
        void LookEast() { look__ = 1; }     // look East (right)
        void LookSouth() { look__ = 2; }    // look South (down)
        void LookWest() { look__ = 3; }     // look West (left)

};

#endif

/* ---------------------------------------------------------------------------------------------------------------
// robot.cpp
// -------------------------------------------------------------------------------------------------------------- */
```

TPP2020-HW1-40847041S 朱自宇 **(請記得修改這裡！未填學號與姓名將會扣很大！若為多人作業，請列出所有成員。)**

```
#include <robot.h>


// move forward
void Robot::GoForward( const int steps )
{
	// North
	if( look__ == 0 )
	{
		posY__ -= steps;
	}
	// East
	else if( look__ == 1 )
	{
		posX__ += steps;
	}
	// South
	else if( look__ == 2 )
	{
		posY__ += steps;
	}
	// West
	else if( look__ == 3 )
	{
		posX__ -= steps;
	}
	else
	{
		printf("GoForward(): Looks at weird direction :(\n");
	}
}



// move backward
void Robot::GoBackward( const int steps )
{
	// North
	if( look__ == 0 )
	{
		posY__ += steps;
	}
	// East
	else if( look__ == 1 )
	{
		posX__ -= steps;
	}
	// South
	else if( look__ == 2 )
	{
		posY__ -= steps;
	}
	// West
	else if( look__ == 3 )
	{
		posX__ += steps;
	}
	else
	{
		printf("GoForward(): Looks at weird direction :(\n");
	}
}


// looking direction
void Robot::LookAt( const int direction )
{
	if ( direction == 0 )
		LookNorth();
	else if ( direction == 1 )
		LookEast();
	else if ( direction == 2 )
		LookSouth();
	else if ( direction == 3 )
		LookWest();
	else
		printf("Look(): invalid input :(\n");
}



/* ------------------------------------------------------------------------------------------------------------------
// main.cpp
// ---------------------------------------------------------------------------------------------------------------*/


#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <stdio.h>
#include <stdlib.h>
#include <maze.h>
#include <robot.h>

using namespace std;
using ull = unsigned long long;

int main()
{
	/*
	*	input
	*/
	int h = 0, w = 0;	// height, width
	ull step = 0;
	scanf("%d %d", &w, &h );
	scanf("%llu", &step );
	getc(stdin);		// get '\n' out

	/* new map
	*/
	Maze NTNU;
	// set up map
	NTNU.setHeight( h );
	NTNU.setWidth( w );
	NTNU.setRoad( '.' );
	NTNU.setObstacle( '#' );
	NTNU.createMap();
	NTNU.setMap();

	// use the map
	const char *m = NTNU.getMap();


	/*
	*	new robot
	*/
	Robot Tcc;

	// since the robot's starting position is 'O', we have to make it to a road '.'
	char map[h][w];
	for( int i = 0; i < h; i+=1 )
	{
		for( int j = 0; j < w; j+=1 )
		{
			if( m[i*w+j] == 'O' )
			{
				map[i][j] = '.';
				Tcc.setposX( j );
				Tcc.setposY( i );
				Tcc.setLook( 0 );
			}
			else
				map[i][j] = m[i*w+j];
		}
	}




	/*
	*	record the path
	*/
	int moveX[200];
	int moveY[200];
	int lookAt[200];
	for( int i = 0; i < 200; i += 1 )// initialize
	{
		moveX[i] = 0;
		moveY[i] = 0;
		lookAt[i] = 0;
	}

	// maybe we don't start a circuit first...
	ull preliminary = 0;
	ull circuit = 0;
	int mode = 1;	// check


	/*
	*	find circuit
	*/
	while(mode)
	{
		// move for 1 step
		Tcc.GoForward(1);
```

```
const int x = Tcc.getposX();
const int y = Tcc.getposY();


// check obstacle for 3 times( turn a round )
for( int i = 0; i < 3; i += 1 )
{
        if( Tcc.getlook() == 0 )
        {
                if( NTNU.isObstacle( x, y-1 ) )
                {
                        Tcc.TurnRight();
                }
        }
        else if( Tcc.getlook() == 1 )
        {
                if( NTNU.isObstacle( x+1, y ) )
                {
                        Tcc.TurnRight();
                }
        }
        else if( Tcc.getlook() == 2 )
        {
                if( NTNU.isObstacle( x , y+1 ) )
                {
                        Tcc.TurnRight();
                }
        }
        else if( Tcc.getlook() == 3 )
        {
                if( NTNU.isObstacle( x-1, y ) )
                {
                        Tcc.TurnRight();
                }
        }
        else
        {
                printf("Tcc looks at weird position :(\n");
        }
}       // end "check obstacle"


// circuit check
for( ull i = 0; i < circuit; i += 1 )
{
        if( ( x==moveX[i]) && (y==moveY[i]) && ( Tcc.getlook()==lookAt[i] ) )
        {
                circuit = circuit - i;
                preliminary = i;
                mode = 0;
        }
}


// check while
if( mode == 0 ) // got the answer :)
{
        break;
}
```

```
        else        // keep going :(
        {
            // record this step
            const int look = Tcc.getlook();
            lookAt[circuit] = look;
            moveX[circuit] = x;
            moveY[circuit] = y;

            circuit += 1;
        }

    }   // end "find circuit"



    ull result;
    if( step == circuit )    // didn't find a circuit :(
    {
        result = step;
    }
    else
    {
        result = ((step-preliminary-1)%circuit)+preliminary;
    }
    printf("%d %d\n", moveX[result], moveY[result] );


    NTNU.freeMap();

    return 0;
}
```

<< 請適當編排以利列印與閱讀，程式碼儘量不要跨行。 >>