```c
#include <stdio.h>

#define N                   25
#define ONE                 1
#define BOOLEAN             unsigned short
#define ABS(x)              (x) * (((2 * (x)) + 1) % 2)
#define ZERO                0

//-----------------------------------------------------------------------------------
//                                              NumBBiggerThenNumA
//                                              ------------------
//
// General      : Checks if the second number (B) is greater than the first number (A).
//
// Parameters   :
//          number1 -  first number (int)
//          number2 -  second number (int)
//
// Return value : If the second number (B) is greater than the first number (A) (BOOLEAN).
//
//-----------------------------------------------------------------------------------
// Programmer : Cohen Idan
// Student No : None
// Date       : 21.10.2019
//-----------------------------------------------------------------------------------
BOOLEAN NumBBiggerThenNumA(int a, int b)
{
    int sub = a - b + ONE;
    BOOLEAN answer = (ONE / (ABS(sub) + (sub) + ONE));
           answer;
}

//-----------------------------------------------------------------------------------
//                                              Exercise 5
//                                              ----------
//
// General : The program checks if the tool can move in other rule.
//
// Input   : 25 numbers, index of player(number), cube result(number).
//
// Process : The program checks the place and the dice roll
//           and then it checks if it has to go back or can go forward.
//
// Output  : The new place of the tool on the board according to the
//           cube roll (unsigned short).
//
//-----------------------------------------------------------------------------------
// Programmer : Cohen Idan
// Student No : 211675038
// Date       : 23.10.2019
//-----------------------------------------------------------------------------------
void main(void)
{
    unsigned short board[N] = { ZERO },
                   start_index_player,
                   cube_result,
                   end_player_index,
                   counter,
                   input;

        (counter = ZERO; counter < N; counter++)
    {
        printf("Enter number for index %ud on borad : ", counter);
        scanf("%ud", &input);
    }

    printf("Enter index of player: ");
    scanf("%ud", &start_index_player);

    printf("Enter cube result: ");
    scanf("%ud", &cube_result);

```

```
73      end_player_index = start_index_player + ONE;
74
75      end_player_index -= NumBBiggerThenNumA(board[end_player_index],
76                                             cube_result) * (cube_result + ONE);
77
78      printf("%hd\n", end_player_index);
79
80  }
```