```
#include <stdio.h>
#include "function.h"

void hanoi(int N, char start, char end, char buf)
{
  if(N>0)
  {
    // 搬上面2個盤子 放到buf(暫存)
    // move the top n-1 disks from start to buf
    hanoi(N-1, start, buf, end);
    // 把底盤從A放到到C
    //printf("%c -> %c\n", start, end);
    //printf("Move the #%d disk from Tower %c to Tower %c\n", N, start, buf);
    printf("%d\n", N);
    // 把上面2個盤子 放到C
    // move the top n-1 disks from buf to end
    hanoi(N-1, buf, end, start);
  }
}


#include<stdio.h>
#include "function.h"

void print(int level, int n)
{
    if (n == 1) {
        printf("%d\n", level);
    }
    else {
        printf("%d ", level);
    }

    if(level<n)
    {
        print(level+1, n);

        if(level==1) {
            printf("%d\n", level);
        }
        else {
            printf("%d ", level);
        }
    }
}


make file
要注意的地方是第四行和第七行文字縮排要用 tab 不能用 space

all: compile

compile:
        gcc main.c -o main

run:
        ./main
```

```
#include <stdio.h>
#define MAP_SIZE 12
#define CAR_SIZE 3
/* #define ONLINE_JUDGE */

int map[MAP_SIZE][MAP_SIZE];
void map_reset(void);
void map_show(void);

int blocks[MAP_SIZE][MAP_SIZE];
void blocks_read(void);
void blocks_put_on_map(void);

int jewels[MAP_SIZE][MAP_SIZE];
void jewels_read(void);
void jewels_put_on_map(void);

int car[CAR_SIZE][CAR_SIZE] = {{'O', 'O', '@'}, {'O', 'O', 'O'}, {'O', 'O', '@'}};
int car_row = 3, car_col = 4;
int car_direction;
int car_earnings;
void car_rotate90(void);
void car_put_on_map(void);
void car_move(void);
int main(void)
{
  int ch;

  blocks_read();
  jewels_read();

    map_reset();
    blocks_put_on_map();
    jewels_put_on_map();
    car_put_on_map();

    #ifndef ONLINE_JUDGE
    map_show();
    #endif

    #ifndef ONLINE_JUDGE
  freopen("actions.txt", "r", stdin);
    #endif

  while ((ch=getchar()) != EOF) {
    if (ch=='R') {
      car_rotate90();
    }

    if (ch=='F') {
      car_move();
    }

    map_reset();
    blocks_put_on_map();
    jewels_put_on_map();
    car_put_on_map();

        #ifndef ONLINE_JUDGE
```

```
    map_show();
      #endif
  }


  printf("Earnings: %d\n", car_earnings);
  printf("Position: row=%d, col=%d\n", car_row, car_col);
  char dirs[] = "RDLU";
  printf("Direction: %c\n", dirs[car_direction]);


  return 0;
}
void blocks_read(void)
{
  int n, i;
  int row, col;
    #ifndef ONLINE_JUDGE
  freopen("blocks.txt", "r", stdin);
    #endif

  scanf("%d", &n);

  for (i=0; i<n; i++) {
    scanf("%d%d", &row, &col);
    blocks[row][col] = 1;
  }
}
void jewels_read(void)
{
  int n, i;
  int row, col;
    #ifndef ONLINE_JUDGE
  freopen("jewels.txt", "r", stdin);
    #endif
  scanf("%d", &n);
  for (i=0; i<n; i++) {
    scanf("%d%d", &row, &col);
    jewels[row][col] = 1;
  }
}
void blocks_put_on_map(void)
{
  int i, j;
  for (i=0; i<MAP_SIZE; i++) {
    for (j=0; j<MAP_SIZE; j++) {
      if (blocks[i][j])
        map[i][j] = '#';
    }
  }
}
void jewels_put_on_map()
{
  int i, j;
  for (i=0; i<MAP_SIZE; i++) {
    for (j=0; j<MAP_SIZE; j++) {
      if (jewels[i][j])
        map[i][j] = '$';
    }
  }
```

```
}
void map_reset(void)
{
  int i, j;
  for (i=0; i<MAP_SIZE; i++) {
    for (j=0; j<MAP_SIZE; j++) {
      map[i][j] = '.';
    }
  }
  for (i=0; i<MAP_SIZE; i++) {
    map[i][0] = 'H';
    map[i][MAP_SIZE-1] = 'H';
  }
  for (j=0; j<MAP_SIZE; j++) {
    map[0][j] = 'H';
    map[MAP_SIZE-1][j] = 'H';
  }
}

void map_show(void)
{
  int i, j;
  for (i=0; i<MAP_SIZE; i++) {
    for (j=0; j<MAP_SIZE; j++) {
      printf("%c", map[i][j]);
    }
    printf("\n");
  }

}
void car_put_on_map(void)
{
  int i, j;
  for (i=0; i<CAR_SIZE; i++) {
    for (j=0; j<CAR_SIZE; j++) {
      map[i+car_row][j+car_col] = car[i][j];
    }
  }
}

void car_rotate90(void)
{
  /* your code */
    int i, j;
    int temp[CAR_SIZE][CAR_SIZE];
    for (i=0; i<CAR_SIZE; i++) {
        for (j=0; j<CAR_SIZE; j++) {
            temp[j][CAR_SIZE-i-1] = car[i][j];
        }
    }
    for (i=0; i<CAR_SIZE; i++) {
        for (j=0; j<CAR_SIZE; j++) {
            car[i][j] = temp[i][j];
        }
    }
    car_direction = (car_direction + 1)%4;
}
```

```
void car_move(void)
{
  /* your code */
    int car_row_try, car_col_try;
    int valid;
    int i, j;

    car_row_try = car_row;
    car_col_try = car_col;

    if (car_direction==0) {
        car_col_try++;
    } else if (car_direction==1) {
        car_row_try++;
    } else if (car_direction==2) {
        car_col_try--;
    } else {
        car_row_try--;
    }

    valid = 1;
    for (i=0; i<CAR_SIZE; i++) {
        for (j=0; j<CAR_SIZE; j++) {
            if (blocks[i+car_row_try][j+car_col_try]!=0
            || map[i+car_row_try][j+car_col_try]=='H') {
                valid = 0;
            }
        }
    }
    if (valid) {
        car_row = car_row_try;
        car_col = car_col_try;
        for (i=0; i<CAR_SIZE; i++) {
            for (j=0; j<CAR_SIZE; j++) {
                if (jewels[i+car_row][j+car_col]>0) {
                    car_earnings++;
                    jewels[i+car_row][j+car_col]--;
                }
            }
        }
    }
}
```