```c
#include <unistd.h>
#include "4-5-check_mate.h"
static int	resolve(char **grid, pos kpos, int size)
{
	int		i = 1;
	while (kpos.y-i >= 0)
	{
		if ((UCELL == 'Q') || (UCELL == 'R'))
			return (1);
		else if ((UCELL == 'B') || (UCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.y+i < size)
	{
		if ((DCELL == 'Q') || (DCELL == 'R'))
			return (1);
		else if ((DCELL == 'B') || (DCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.x-i >= 0)
	{
		if ((LCELL == 'Q') || (LCELL == 'R'))
			return (1);
		else if ((LCELL == 'B') || (LCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.x+i < size)
	{
		if ((RCELL == 'Q') || (RCELL == 'R'))
			return (1);
		else if ((RCELL == 'B') || (RCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.y-i >= 0 && kpos.x-i >= 0)
	{
		if ((ULCELL == 'Q') || (ULCELL == 'B'))
			return (1);
		else if ((ULCELL == 'R') || (ULCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.y-i >= 0 && kpos.x+i < size)
	{
		if ((URCELL == 'Q') || (URCELL == 'B'))
			return (1);
		else if ((URCELL == 'R') || (URCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.y+i < size && kpos.x+i < size)
	{
		if ((i == 1) && (DRCELL == 'P'))
			return (1);
		if ((DRCELL == 'Q') || (DRCELL == 'B'))
			return (1);
		else if ((DRCELL == 'R') || (DRCELL == 'P'))
			break;
		i++;
	}
	i = 1;
	while (kpos.y+i < size && kpos.x-i >= 0)
	{
		if ((i == 1) && (DLCELL == 'P'))
			return (1);
		if ((DLCELL == 'Q') || (DLCELL == 'B'))
			return (1);
		else if ((DLCELL == 'R') || (DLCELL == 'P'))
			break;
		i++;
	}
	return (0);
}
```

```c
static void	find_king(char **grid, pos *kpos)
{
	int	x;
	int	y;

	y = 0;
	while (*(grid + y))
	{
		x = 0;
		while (*(*(grid + y) + x))
		{
			if (*(*(grid + y) + x) == 'K')
			{
				kpos->x = x;
				kpos->y = y;
				return ;
			}
			x++;
		}
		y++;
	}
}

static int	check_mate(char **grid, int size)
{
	pos	kpos;

	find_king(grid, &kpos);
	if (resolve(grid, kpos, size))
		return (1);
	return (0);
}

int	main(int argc, char **argv)
{
	int	i;

	if (argc > 1)
	{
		i = 0;
		while (*(argv + i + 1))
		{
			*(argv + i) = *(argv + i + 1);
			i++;
		}
		*(argv + i) = NULL;
		i = 0;
//		while (*(argv + i))
//		{
//			write(1, *(argv + i), 4);
//			write(1, "\n", 1);
//			i++;
//		}
		check_mate(argv, argc - 1) ? write(1, "Success", 7) : write(1, "Fail", 4);
	}
	write(1, "\n", 1);
}
```