Data Structure Programming Assignment #1

(Submit solution to my e-mail: utilForever@gmail.com)

1. 수학 분야에서 'random walk'라고 하는 오랫동안 관심의 대상이 되었던 문제들이 많이 있다. 이러한 문제 중 아주 간단한 것을 제외하고는 모두 해결하기 매우 어려우며 대부분 해결되지 못하고 남아 있다. 예로 다음의 문제를 생각해 보자.

   한 (술 취한) 바퀴벌레가 $n \times m$ 크기의 직사각형 방 중간의 한 타일 위에 있다. 바퀴벌레는 임의로 타일에서 타일로 걸어간다. 바퀴벌레가 현재 타일에서 그 주위 8 개의 타일(벽 옆에 있지 않은 경우)로 **같은 확률을 가지고** 움직인다고 가정하자. 이때, 적어도 한 번씩 방의 모든 타일을 지나가는데 걸리는 시간은 얼마인가?

   순수한 확률 이론 기법에 의해 이런 문제를 해결하는 것은 어렵지만, 컴퓨터를 이용하면 매우 쉽게 해결된다. 이러한 기법을 '시뮬레이션(Simulation)'이라고 한다. 이 기법은 교통 흐름, 재고 관리 등을 예측하는데 광범위하게 사용되고 있다. 위의 문제는 다음과 같은 방법을 이용해 표현할 수 있다.

   $n \times m$ 배열 $count$는 방에서 바퀴벌레가 각 타일을 방문하는 횟수를 나타내는데 사용한다. 이 배열의 모든 원소의 초기값은 0 이다. 방에서 바퀴벌레의 위치는 좌표 $(ibug, jbug)$로 나타낸다. 바퀴벌레의 8 가지 가능한 이동은 타일들의 위치, 즉 $(ibug + imove[k], jbug + jmove[k])$로 나타내며, 여기서 k는 $0 \leq k \leq 7$이고 $imove$와 $jmove$는 다음과 같다.

$imove[0] =$	-1	$jmove[0] =$	1
$imove[1] =$	0	$jmove[1] =$	1
$imove[2] =$	1	$jmove[2] =$	1
$imove[3] =$	1	$jmove[3] =$	0
$imove[4] =$	1	$jmove[4] =$	-1
$imove[5] =$	0	$jmove[5] =$	-1
$imove[6] =$	-1	$jmove[6] =$	-1
$imove[7] =$	-1	$jmove[7] =$	0

   주어진 정방의 8 개 타일 중 어느 하나로 임의로 움직이는 random walk 는 0 과 7 사이의 임의값 k를 생성해 시뮬레이션을 수행할 수 있다. 물론 바퀴벌레는 방 밖으로 벗어나 움직일 수 없으며, 벽 위로 움직이는 좌표는 무시되어야 하고 새로운 임의 조합을 통해 움직임이 형성된다. 한 타일에 들어갈 때마다

해당하는 배열 값이 증가되며 이것은 지금까지 바퀴벌레가 그 타일을 방문한 횟수를 나타내는 것이다. 모든 타일에 적어도 한 번은 들어갔을 때 이 실험은 종료된다.

이와 같은 시뮬레이션 실험을 수행하는 프로그램을 작성하라. 이 프로그램은 다음 조건들을 가진다.

a. m과 n의 값은 $2 \leq n \leq 40$, $2 \leq m \leq 20$으로 조절한다.
b. 다음과 같은 경우에 대해 실험한다.
    i. $n = 15, m = 15$, 시작점 : (9, 9)
    ii. $n = 39, m = 19$, 시작점 : (0, 0)
c. 반복 실행의 한계를 가진다. 즉, 실험 중 바퀴벌레가 들어갈 수 있는 타일의 수의 최댓값을 정하라. 이 가정은 프로그램이 종료될 수 있도록 하기 위한 목적이다. 최대 50,000 으로 결정하는 것이 이 연습문제에 적당하다.
d. 각 실험에 대해 다음 결과를 출력한다.
    i. 적법한 바퀴벌레의 총 이동 수
    ii. 마지막 $count$ 배열(이것은 움직임에 대한 밀도, 즉 방에서 움직임을 통해 각 타일에 넣게 되는 횟수를 나타낸다)

2. 체스는 게임 자체에 독립적으로 매우 흥미있는 오락적인 면을 갖고 있다. 이 중 많은 것들은 기사(Knight)의 L 형태의 움직임에 기본을 두고 있다. 대표적인 예가 18 세기 초반부터 많은 수학자와 게임 분석가들의 흥미를 집중시켰던 기사의 여행(Knight's Tour)에 관한 문제다. 간단히 말해서, 주어진 체스판의 임의의 위치에서 출발해 각 위치를 오직 한 번만 방문하면서 모두 64 개의 위치를 방문하는 문제다. 방문하는 위치의 순서를 나타내기 위해 체스판의 각 위치에 1, 2, …, 64 의 번호를 부여한다. 기사는 마지막에 한 번 더 움직여서 처음 위치로 돌아와야 할 필요는 없다. 이것이 가능한 경우의 기사 여행을 반복 가능이라 고 한다. 기사 여행의 해결책 중 가장 천재적인 것은 1823 년 J.C. Warnsdorff 가 제시한 것이다. 그가 사용한 규칙은 기사는 항상 아직 방문되지 않은 위치로 나가는 출구가 가장 적은 위치 중의 하나로 이동 하는 것이다.

   이와 같은 유형의 문제를 풀기 위해 결정해야 할 가장 중요한 사항은 데이터를 어떤 방법으로 컴퓨터에 표현할 것인가 하는 점이다. 아마도 아래 그림과 같이 체스판을 8×8 배열 $board$로 나타내는 것이 가장 자연스러운 것이다. 이 그림에서는 기사가 (4, 2) 위치에서 이동 가능한 8 가지 방향을 표시했다. 일반 적으로 (i, j) 위치에서 기사가 움직일 수 있는 위치는 $(i-2, j+1)$, $(i-1, j+2)$, $(i+1, j+2)$, $(i+2, j+1)$, $(i+2, j-1)$, $(i+1, j-2)$, $(i-1, j-2)$ 및 $(i-2, j-1)$ 등이다. 그러나 기사의 위치 (i, j)가 체스판의 경계에 근접해 있을 경우는 위의 가능 위치 중 체스판을 벗어나는 것은 허용되지 않는다. 위의 8 가지 방향은 아래와 같이 $ktmov1$과 $ktmov2$를 이용해 효과적으로 나타낼 수 있다.

	0	1	2	3	4	5	6	7
0								
1								
2		8		1				
3	7				2			
4			K					
5	6				3			
6		5		4				
7								

$ktmov1$	$ktmov2$
-2	1
-1	2
1	2
2	1

2	-1
1	-2
-1	-2
-2	-1

이와 같은 표현법을 이용해 (i, j) 위치에 있는 기사가 움직일 수 있는 위치는 $(i + ktmov1[k], j + ktmov2[k])$이다. 단, 이동된 위치가 체스판 위에 있어야 하고 k는 0 과 7 사이의 값이어야 한다.

다음은 Warnsdorff 의 규칙을 이용해 기사 여행 문제의 해를 구하는 알고리즘을 기술한 것이다.

a. [**체스판의 초기화**] $0 \le i, j \le 7$에 대해, $board[i][j] = 0$으로 설정한다.
b. [**시작점 설정**] (i, j)를 읽고 출력한 다음 $board[i][j] = 1$로 설정한다.
c. [**반복**] $2 \le m \le 64$에 대해, (d)부터 (g)까지의 과정을 수행한다.
d. [**다음으로 이동 가능한 위치 파악**] (i, j) 위에서 기사가 움직일 수 있는 방향 중에서 어느 위치로 이동할 수 있는가를 조사해, 이동 가능한 위치에 대한 리스트 $(nexti[l], nextj[l])$을 작성한다. 변수 $npos$는 가능한 상태 수다. [즉, 이 단계를 수행하고 나면 0 과 7 사이의 값을 갖는 임의의 위치 k에 대해 $next[l] = i + ktmov1[k]$이고 $nextj[l] = j + ktmov2[k]$이 된다. 다음 이동 위치 $(i + ktmov1[k], j + ktmov2[k])$ 중에는, 체스판의 경계를 벗어나거나 또는 이미 기사가 방문했던 위치, 즉 0 이 아닌 값을 갖기 때문에 이동 가능한 위치에서 제외되는 경우가 있다. 또한 변수 $npos$는 0 과 8 사이의 값을 갖는다.]
e. [**특별한 경우의 검사**] 만약 $npos = 0$이면 기사의 여행은 즉시 종료된다. 실패를 보고한 다음, 단계 (h)로 간다. 만약 $npos = 1$이면 다음으로 움직일 수 있는 경우가 한 가지 밖에 없다. 즉, $min = 0$으로 하고 단계 (g)로 간다.
f. [**최소의 출구를 가진 다음 위치 탐색**] $1 \le l \le npos$를 만족하는 l에 대해, $exits[l]$에 위치 $(nexti[l], nextj[l])$로부터 출구의 개수를 지정한다. 즉, 모든 l에 대해 다음 위치 $(nexti[l] + ktmov1[k], nextj[l] + ktmov2[k])$가 $(nexti[l], nextj[l])$의 출구인가를 확인하고 $exits[l]$에 출구의 갯수를 저장한다(출구란 체스판 위에 있으면서 아직 기사에 의해 방문되지 않은 위치다). 마지막으로 배열 $exits$에서 최솟값을 갖는 원소의 인덱스를 변수 min에 할당한다.(배열 $exits$에서 최솟값을 갖는 원소가 하나 이상 존재할 경우에는 첫 번째 값을 min에 할당한다. 이렇게 해서 해를 구한다는 보장은 없으나 해를 구할 확률이 상당히 높아지게 되며 이 문제의 목적에 충분하게 된다.)
g. [**기사의 이동**] $i = nexti[min]$,. $j = nextj[min]$, 그리고 $board[i][j] = m$을 할당한다. 즉, (i, j)는 기사의 새로운 위치를 나타내며 $board[i][j]$는 적절한 순서로 이동하는 것을 기록한다.
h. [**출력**] 기사 여행 결과를 보여 주는 $board$를 출력하고, 알고리즘을 종료한다.

위의 알고리즘을 C++ 프로그램으로 작성하라.