Assignment #3, Advanced Programming (2016 Spring Semester)

Due date: 2016/05/27, 23:59:59

1. 키보드로부터 입력받은 글자 및 숫자들을 리스트에 저장하는 프로그램을 구현하라. 입력받는 문자는 띄어쓰기나 개행으로 구분되며, 하나의 노드로 만들어 리스트의 맨 앞에 삽입된다. 입력된 데이터가 리스트에 저장될 때 마다 리스트의 상태를 출력한다.

입력 예제	리스트 상태
a t 3 d 21 o 3000 k	a t -> a 3 -> t -> a d -> 3 -> t -> a 21 -> d -> 3 -> t -> a o -> 21 -> d -> 3 -> t -> a 3000 -> o -> 21 -> d -> 3 -> t -> a k -> 3000 -> o -> 21 -> d -> 3 -> t -> a

2. 입력 데이터의 타입을 구분하는 프로그램을 구현하라. 사용자는 문자열, 정수, 실수로 총 세가지 형태의 데이터를 입력하며, 프로그램은 먼저 입력 데이터를 구분한 뒤, 클래스 템플릿을 사용하여 데이터를 저장한다. 저장된 데이터는 타입에 따라 문자열, 정수, 실수의 리스트에 오름차순으로 저장된다. 프로그램은 'END'가 입력되면 리스트에 저장되어있는 데이터를 문자열, 정수, 실수의 순서로 출력하고 프로그램을 종료한다. 각 데이터는 띄어쓰기나 개행으로 구분한다.

입력 예제	출력 예제
A 1 B 12 6.5 5.1 CD 1596 135.5 END	문자열 : A B CD 정수 : 1 12 1596 실수 : 5.1 6.5 135.5

3. 파일 내에 저장된 단어들을 오름차순으로 정렬한 후 두 개의 2차원 연결 리스트에 저장하고 오름차순으로 정렬하는 프로그램을 구현하라. 프로그램은 'input.txt' 파일을 읽어 각 라인 마다 포함되어 있는 정수 및 문자열 데이터를 타입에 따라 각각 2차원 연결 리스트에 저장한다. 이 때, 정수 또는 문자열 데이터가 없는 라인에 대해서는 노드를 생성하지 않는다. 각 데이터들은 템플릿 노드에 각자의 데이터 형으로 저장하고, 각 리스트에서 오름차순 정렬을 수행한다. 이후 각 리스트는 각 리스트가 가진 첫 노드들끼리 비교하여 리스트 단위로 상하방향 오름차순 정렬을 수행한다. 단, 문자열과 정수가 붙어있는 경우에는 문자열로 취급한다. 정렬이 끝난 두 개의 리스트는 'result.txt'에 정수 리스트, 문자열 리스트 순으로 출력된다.

초기 파일	정렬된 파일
ff aa 2 5 cc dd 4 3 zz ee bb 1 6	1 6 aa ff 2 5 bb ee 3 4 cc dd zz

4. 지하철 경로를 찾는 프로그램을 구현하라. 프로그램은 특정 호선의 일부의 정보를 갖는 파일을 읽어 각 지하철 노선을 더블 연결 리스트로 만든다. 각 리스트를 구성하는 노드들은 해당 지하철 노선에 포함되는 역에 대한 정보를 갖는다. 사용자로부터 출발 역과 도착 역의 이름을 입력 받아 출발 역에서부터 도착 역까지의 지하철 경로를 출력하라. 입력 파일은 아래와 같다.

입력 파일	
까치산 신정네거리 신정네거리 양천구청 양천구청 도림천 도림천 신도림 신도림 문래 문래 영등포구청 영등포구청 당산 당산 합정 합정 홍대입구 홍대입구 신촌 신촌 이대 이대 아현 아현 충정로	

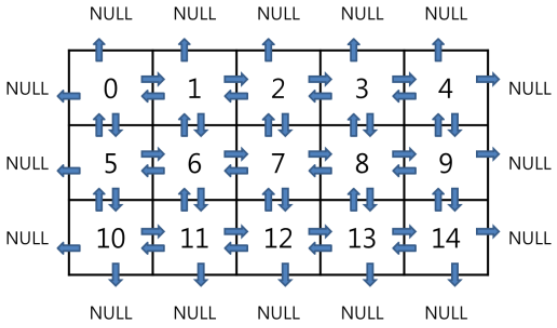
입력 예제	출력 예제
출발지 : 신정네거리 도착지 : 신도림	신정네거리 -> 양천구청 -> 도림천 -> 신도림

입력 예제 2	출력 예제
출발지 : 영등포구청 도착지 : 까치산	영등포구청 -> 문래 -> 신도림 -> 도림천 -> 양천구청 -> 신정네거리 -> 까치산

5. 파일을 읽어 숫자퍼즐을 구현하라. ‘puzzle.info’ 파일에는 퍼즐의 크기와 퍼즐을 섞는 방법이 기록되어 있고, 퍼즐의 크기는 $n \times m$ (n행, m열) 형태로 기록이 되어있다.

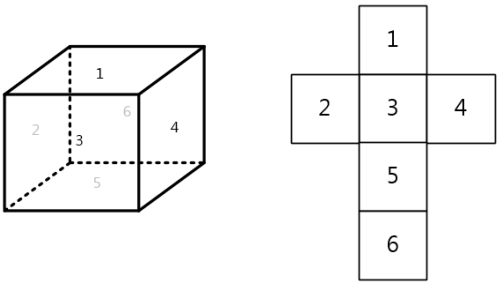
입력 파일 예제
3X5 drrrddrruullllldrrruull

프로그램이 처음 시작될 때 퍼즐의 크기만큼 양방향 2차원 연결 리스트를 만들고, 1행 1열부터 n행 m열까지 첫 줄부터 왼쪽에서 오른쪽 방향으로 0부터 1씩 증가시키며 초기화 시킨다. 양방향 2차원 연결 리스트 구현 시 노드 타입 2차원 배열을 사용하지 않도록 한다.



퍼즐을 섞어 문제를 만들 때는 파일로부터 이동 명령어를 입력 받으며, 만들어진 퍼즐 문제를 풀 때에는 사용자로부터 이동 명령어를 입력 받아 0값이 저장된 노드를 상하좌우 방향으로 이웃 노드들의 위치와 교환한다. 여기서 사용되는 이동 명령어로는 'u(up), d(down), l(left), r(right)' 네 가지가 존재한다. 노드의 이동은 숫자 퍼즐 범위 밖으로 벗어나서는 안되고, 만약 벗어나는 입력이 들어올 경우 그 입력을 무시하게 된다. 퍼즐은 초기 상태로 되돌아가면 완성되며, 퍼즐이 완성되면 프로그램은 종료된다.

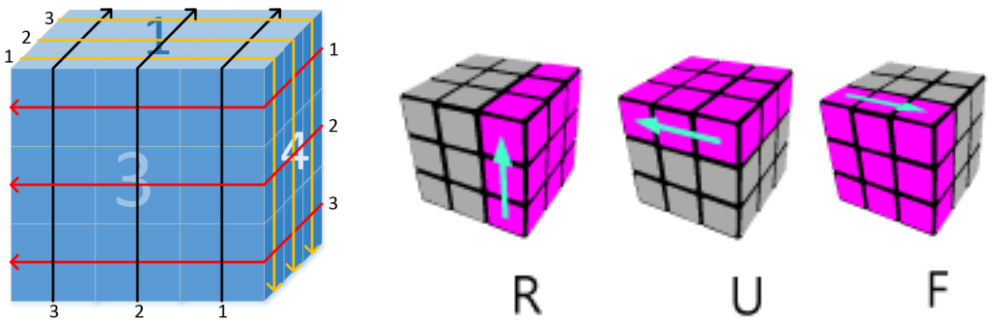
6. 원형 연결 리스트를 이용하여 3x3 정육면체 큐브를 구현하라.



큐브의 초기 상태는 다음과 같이 설정한다.

1면	2면	3면
11 12 13 14 15 16 17 18 19	21 22 23 24 25 36 27 28 29	31 32 33 34 35 36 37 38 39
4면	5면	6면
41 42 43 44 45 46 47 48 49	51 52 53 54 55 56 57 58 59	61 62 63 64 65 66 67 68 69

큐브는 다음과 같이 각 x, y, z축에 대해 수직인 방향의 원형 연결 리스트를 3개씩 갖는다.



각 리스트를 구성하는 노드들은 큐브의 블록이다. 프로그램은 큐브의 동작이 저장 되어있는 파일을 읽어 큐브 동작을 수행하고, 수행이 완료된 큐브의 상태를 출력한다. 큐브의 동작에는 'R', 'U', 'F' 세가지가 있으며, 각 동작은 면 '3'을 기준으로 동작한다. 'R'은 위로(검정색 방향), 'U'는 왼쪽으로(빨간색 방향), 'F'는 오른쪽으로(노란색 방향) 회전시키며, 이 때 회전은 시계방향으로 90° 회전을 기본으로 한다. 큐브의 동작은 다음과 같이 '[위치][동작]'의 형식으로 주어진다. 예를 들어 2R일 경우 검정색 2번 리스트에 포함되는 블록들을 90° 회전시키는 동작이고, 3U일 경우 빨간색 3번 리스트에 포함되는 블록들을 회전시키는 동작이다

입력 파일 예제	출력 예제	
큐브 동작 1R	1면 11 12 **33** 14 15 **36** 17 18 **39** 2면 21 22 23 24 25 26 27 28 29 3면 31 32 **53** 34 35 **56** 37 38 **59**	4면 **47 44 41** **48** 45 **42** **49 46 43** 5면 51 52 **63** 54 55 **66** 57 58 **69** 6면 61 62 **13** 64 65 **16** 67 68 **19**

7. 파일에 수입 지출 정보를 관리하는 가계부 프로그램을 구현하라. 이 프로그램은 *Load, Save, Insert, Calculate, Exit* 의 기능을 수행할 수 있다. *Load*는 'acbook.txt' 파일에 저장된 수입, 지출 항목과 그에 대한 금액을 각각 노드에 기록한 뒤 Queue에 저장한다. *Save*는 Queue에 남아있는 항목들을 'acbook.txt' 파일에 새로 저장한다. 또한, *Insert*는 사용자가 키보드로 입력한 income 혹은 expend 와 그에 대한 금액 정보를 Queue에 저장하고, *Calculate*는 Queue에 존재하는 모든 노드들을 pop을 통해 꺼내면서 '+' 연산자 오버로딩을 이용하여 수입/지출 값을 차례대로 더해 하나의 노드가 존재하는 Queue로 만드는 기능을 수행한다. 이 때, 합산 금액이 0보다 클 경우 'income', 작을 경우 'expend'로 표시하고 합산 금액과 같이 저장한다. 또한 연산 도중 수입이 0 미만으로 떨어지면 경고를 출력한다. *Exit* 메뉴를 통해 프로그램을 종료할 수 있다.

입력 파일 예제	출력 예제	출력 파일 예제
income 3200000 expend 450000 expend 87000 expend 62000 income 100000 expend 18000	1.Load 2.Save 3.Insert 4.Calculate 5.Exit 1 Load complete 4 Calculate success 3 expend 100000 4 Calculate success 2 Save complete	income 2583000

8. 학생들의 정보를 관리하는 프로그램을 구현하라. 프로그램은 '추가', '제거', '출력' 연산을 수행할 수 있다. '추가' 연산은 학생의 정보를 사용자로부터 입력 받아 노드를 생성하고, 생성된 노드를 리스트에 추가하는 동작을 수행한다. 리스트에 노드를 추가하는 동작은 '+' 연산자 오버로딩을 이용하여 수행하도록 한다. 이때, 입력되는 학생 정보에는 동명인이 있을 수 있으며, 학번이 같은 경우는 존재하지 않는다. '제거' 연산은 사용자로부터 입력 받은 학생의 이름과 학번을 이용하여 리스트에서 노드를 찾고, 해당 노드가 있을 경우 리스트에서 삭제하는 동작을 수행한다. 마찬가지로 해당 동작 또한 '-' 연산자 오버로딩을 이용하여 수행하도록 한다. '출력' 연산의 경우 학생들의 학번 혹은 이름을 기준으로 오름차순으로 출력 한다. 단, 이름을 기준으로 출력할 경우 동명인의 출력 순서는 중요하지 않다.

출력 예제

```
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 1
------------------------------
이름 : 장승철
학번 : 2013722045
성별 : 여
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 1
------------------------------
이름 : 박준택
학번 : 2013722092
성별 : 남
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 1
------------------------------
이름 : 박준택
학번 : 2013722053
성별 : 여
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 3
------------------------------
메뉴 (0:종료 1:이름 2:학번)
입력 : 1
------------------------------
이름 : 박준택
학번 : 2013722092
성별 : 남

이름 : 박준택
학번 : 2013722053
성별 : 여

이름 : 장승철
학번 : 2013722045
성별 : 여
==================
```

```
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 2
------------------------------
이름 : 박준택
학번 : 2013722092
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 3
------------------------------
메뉴 (0:종료 1:이름 2:학번)
입력 : 2
------------------------------
이름 : 장승철
학번 : 2013722045
성별 : 여

이름 : 박준택
학번 : 2013722053
성별 : 여
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 2
------------------------------
이름 : 박준택
학번 : 2013722092
------------------------------
학생 정보가 존재하지 않습니다.
==================
메뉴 (0:종료 1:추가 2:제거 3:출력)
입력 : 0
------------------------------
종료
==================
```

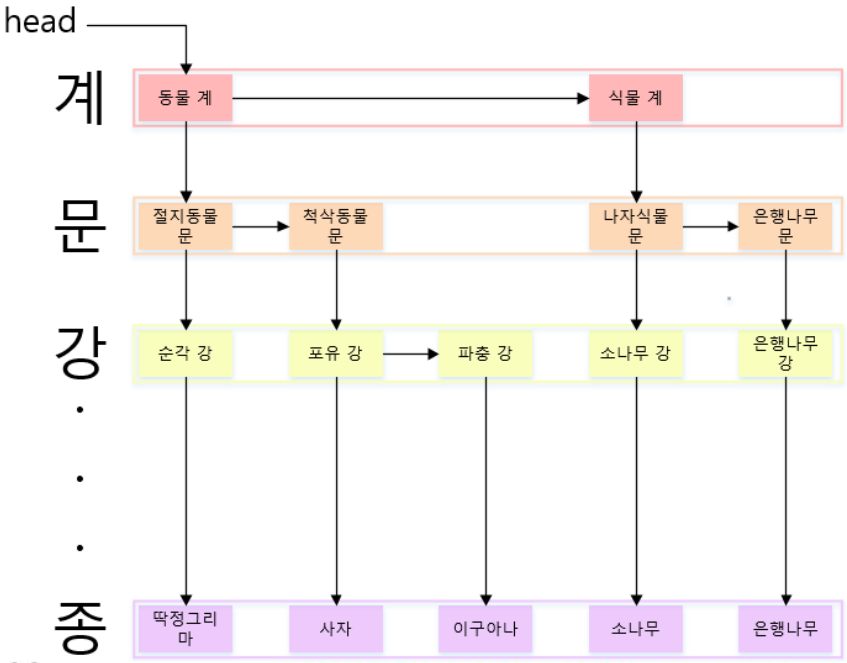
9. 파일에 기록된 1차 방정식을 읽어 미지수 x를 구하는 프로그램을 구현하라. "equation.txt" 파일에 기록된 방정식의 숫자와 부호들을 노드에 기록한 뒤 리스트에 저장한다. 입력될 수 있는 부호는 '+', '-', '*', '/', '(', ')', '=' 와 미지수 'x' 가 있고, 미지수를 포함한 모든 부호는 좌항과 우항 관계없이 여러 개 존재할 수 있다. 단, 여기서 다항식은 고려하지 않는다. 프로그램은 전체 식을 저장하는 하나의 리스트 또는 좌항과 우항을 각각 저장하는 두 개의 리스트를 가질 수 있다. 연산 순서는 괄호 안에 있는 식을 최우선으로 계산하고 곱셈과 나눗셈은 덧셈과 뺄셈보다 먼저 계산이 되어야 한다. 입력 파일로부터 읽어온 방정식을 계산하여 찾은 미지수 x의 해를 출력한 후 프로그램이 종료된다.

입력 파일 예제	출력 예제
5*(3+16*x)=x/4	x = -0.188088

10. 파일로부터 입력 받은 문자들을 저장하는 Queue와 Stack을 구현하라. Queue와 Stack은 push 기능이 구현된 연결 리스트를 상속받아 사용하며, pop 기능은 각 자료구조에 따로 존재한다. 본 프로그램은 'inheritance.txt' 파일에 저장된 문자열을 한 글자씩 받아와 Queue와 Stack에 저장한다. 입력 파일 내에 존재하는 모든 문자열을 읽으면 Queue와 Stack에 저장된 모든 데이터를 꺼내어 출력한다.

입력 파일 예제	출력 예제
Hello!	Queue : Hello! Stack : !olleH

11. 동물의 정보를 관리하는 프로그램을 구현하라. 생물 분류학에서는 생물을 특정 기준 ('계', '문', '강', '목', '과', '속', '종')들에 따라 분류한다. 예를 들어 인간의 생물학적 분류의 경우 '동물계-척삭동물문-포유강-영장목-사람과-사람속-사람' 이 된다. 해당 프로그램은 '계', '문', '강', '목', '과', '속', '종'의 분류 기준들의 정보를 다음과 같이 구성한다. 각각의 분류 기준들은 해당하는 항목들과 하위 분류 기준의 리스트 정보를 갖는다. '계' 클래스는 해당 기준에 포함되는 항목의 정보와 하위 기준인 '문' 클래스에 해당하는 노드들의 리스트를 갖는다. '문' 노드는 '문' 항목의 정보와 '강' 노드 리스트를 가지며, '강', '목', '과', '속' 노드도 같은 형식으로 구성된다. '종' 클래스의 경우에는 해당 하는 동물의 이름과 전체 분류 정보를 갖는다. 이에 따라 프로그램이 구성하는 전체적인 구조의 예는 다음과 같다.



프로그램은 동물의 분류 정보가 저장되어 있는 파일을 읽어 동물 분류 정보들을 각각 노드로 만들고, 해당되는 기준의 리스트에 추가한다. 사용자는 해당 프로그램을 이용하여 특정 동물의 분류 정보나 특정 기준("계", "문", "강", "목", "과", "속", "종")에 해당하는 항목들의 정보를 확인할 수 있다.

입력 파일 예제	출력 예제
-사자 동물계 척삭동물문 포유강 식육목 고양이과 표범속 사자 -사람 동물계 척삭동물문 포유강 영장목 사람과 사람속 사람 -북극곰 동물계 척삭동물문 포유강 식육목 곰과 큰곰속 북극곰	메뉴 (0:종료 1:항목 선택 2:검색) 입력 : 2 ---------- 동물 이름 : 사자 ---------- 사자는 동물계 – 척상동물문 – 포유강 – 식육목 – 고양이과 – 표범속 – 사자(이)다. ========== 메뉴 (0:종료 1:항목 선택 2:검색) 입력 : 1 ---------- 메뉴 (0:종료 1:계 2:문 3: 강 4:목 5:과 6:속 7:종) 입력 : 4 ---------- [과] 1. 식육목 2. 영장목 메뉴 (0:종료 1:식육목 2:영장목) 입력 : 1 ---------- [식육목] 1. 고양이과 2. 곰과 메뉴 (0:종료 1:고양이과 2:곰과) 입력 : 0 ========== 메뉴 (0:종료 1:항목 선택 2:검색) 입력 : 0 ---------- 종료

12. [프로젝트 연계 문제] 아래의 설명을 참고하여 2048 게임을 구현하시오.

2048 게임은 4×4 블록을 기반으로 수행된다. 본 문제에서 구현해야 하는 2048 게임은 그림 1과 같이 총 16개의 블록(block)으로 구성된 보드(board) 위에서 실행되며, 한 블록은 숫자 값(number)과 좌표 값(x, y)를 갖는다. 그림 2는 보드를 구성하는 각 블록의 좌표 값을 나타낸다.

512		2	16
2		8	
	4		
	32		2048

그림 1. 2048 게임 보드의 예

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

그림 2. 각 block의 좌표 값

본 문제에서 2048 게임은 아래의 **MyBlock** class와 **MyBoard** class를 사용하여 구현한다. **MyBlock** class의 첫 번째 멤버 변수 m_number는 각 블록이 가지는 숫자 값을 나타내는 변수이며, 두 번째와 세 번째 멤버 변수인 m_x, m_y는 해당 블록의 x, y 좌표 값을 저장하는 변수이다. 또한, **MyBlock** class는 네 개의 MyBlock 포인터를 가지고 있으며, m_pUp, m_pDown, m_pLeft, m_pRight는 각각 블록의 상, 하, 좌, 우에 위치하는 블록을 가리키는 포인터로 사용된다.

MyBoard class는 16개의 블록 중 (0, 0)의 좌표 값을 가지는 블록을 가리키는 멤버 변수 m_pHead를 가지고 있으며, 사용자의 입력에 따라 동작되는 Up, Down, Left, Right의 네 가지 멤버 함수를 가지고 있다. 해당 함수들의 동작 방식은 아래에 자세하게 기술되어 있다.

프로그램 실행 시, main 함수에서 **MyBoard** class의 객체를 생성하고, 객체 생성에 따라 **MyBoard** class의 생성자에서는 **MyBlock** class 객체 16개를 생성한 후 각 블록이 가지는 좌표 값(m_x, m_y)를 그림 2와 같이 초기화한다. 각 블록의 숫자 값은 2048 게임 규칙에 따라 랜덤으로 생성되고 각 객체의 멤버 변수 m_number에 초기화되며, 이 후의 2048 게임의 동작 방식은 아래를 참고한다.

```
class MyBlock
{
   private:
      int   m_number, m_x, m_y;
      MyBlock*   m_pUp;
      MyBlock*   m_pDown;
      MyBlock*   m_pLeft;
      MyBlock*   m_pRight;

   public:
      MyBlock();
      ~ MyBlock();
      …
}
```

```
class MyBoard
{
   private:
      MyBlock*   m_pHead;

   public:
      MyBoard();
      ~MyBoard();

      bool Up();
      bool Down();
      bool Left();
      bool Right();
      …
}
```

```
void main()
{

   MyBoard   m_Board;

   …
}
```

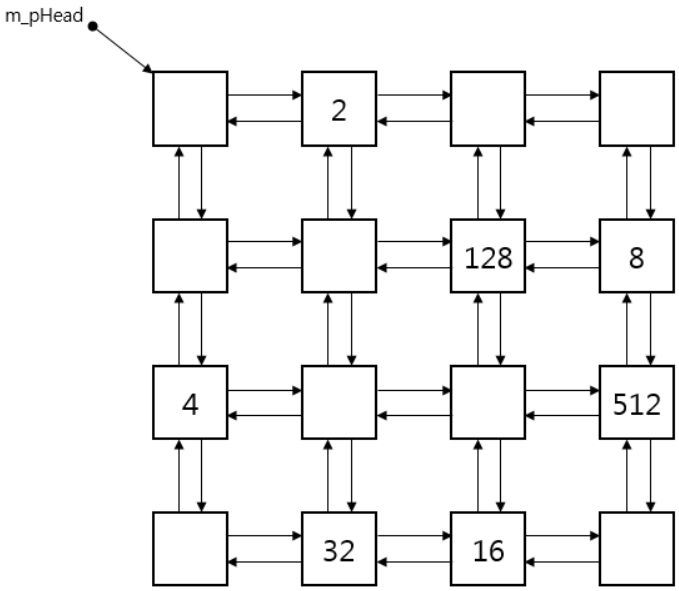


그림 3. MyBoardBlockList 구성도

[2048 게임 동작 방법]

본 문제에서 구현하는 2048 게임은 16개의 블록을 가진 4×4 보드에서 게임이 이뤄진다.

게임 방식은 다음과 같다. 매 턴이 시작하기 전, 랜덤한 위치의 빈 블록에 숫자 값 2가 생성되고, 상(1), 하(2), 좌(3), 우(4) 중 하나의 입력을 받는다(그림 4 참조). 입력을 받은 후에는 각 입력에 따라 MyBoard의 멤버 함수인 Up, Down, Left, Right의 함수가 수행되는데, 이 함수에서는 모든 블록이 입력 받은 방향의 가장 끝에 위치한 빈 블록으로의 이동이 수행된다. 이 때 이동하는 방향에서 같은 값을 가진 두 블록이 만나게 되는 경우에는 두 블록이 가진 숫자를 더하여 이동한 방향의 유효한(비어 있는 블록의) 가장 끝자리에 하나의 블록으로 위치하게 된다. 이동의 우선순위는 입력 받은 방향에 위치할수록 높다. 2048의 값을 가진 블록이 생성되면 게임에서 승리하게 되며 빈 블록이 없고 더 이상 합칠 수 있는 블록이 없다면 게임에서 지게 된다.

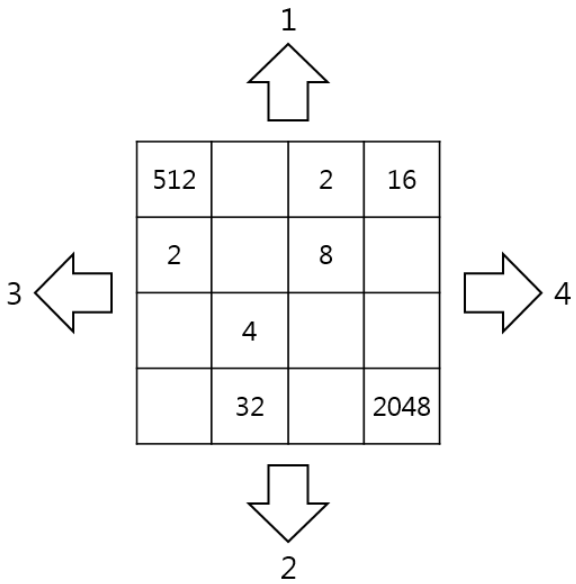


그림 4. 블록 이동 방향에 대한 키 값

[프로그램의 입력 및 출력]

본 프로그램은 사용자의 입력에 따라 블록의 이동과 프로그램의 종료를 수행한다. 1부터 4까지의 정수 중 하나가 입력되는 경우, 프로그램은 이를 이동 방향에 대한 키 값으로 해석하여 각 블록들을 키 값의 방향으로 이동시킨다. 승리 조건이나 패배 조건이 성립하거나 -1이 입력 되면 프로그램은 종료된다.

다음은 입력 및 출력 대한 예시이다.

Ex 1)

```
...
[Board]
[  4    _    2    _  ]
[  512  _    _    _  ]
[  512  512  _    _  ]
[  512  512  _    512]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 3

[Board]
[  4    2    _    _  ]
[  512  _    _    _  ]
[  1024 _    _    2  ]
[  1024 512  _    _  ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 2

[Board]
[  _    _    _    _  ]
[  4    _    _    _  ]
[  512  2    _    _  ]
[  2048 512  _    2  ]

Win!!
[program exit]
```

Ex 2)

```
...
[Board]
[  512  1024  256  128 ]
[  4    32    2    16  ]
[  8    4     64   32  ]
[  16   _     128  _   ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 3

[Board]
[  512  1024  256  128 ]
[  4    32    2    16  ]
[  8    4     64   32  ]
[  16   128   2    _   ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 4

[Board]
[  512  1024  256  128 ]
[  4    32    2    16  ]
[  8    4     64   32  ]
[  2    16    128  2   ]

Lose!!
[program exit]
```

Ex 3)

```
[Start]
[Board]
[  _    _    _    2  ]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  _    _    _    _  ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 3

[Board]
[  2    _    2    _  ]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  _    _    _    _  ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 2

[Board]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  2    _    2    2  ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> 3

[Board]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  _    _    _    _  ]
[  4    2    _    _  ]

[Menu]
1.Up  2.Down  3.Left  4.Right
Input >> -1

[program exit]
```