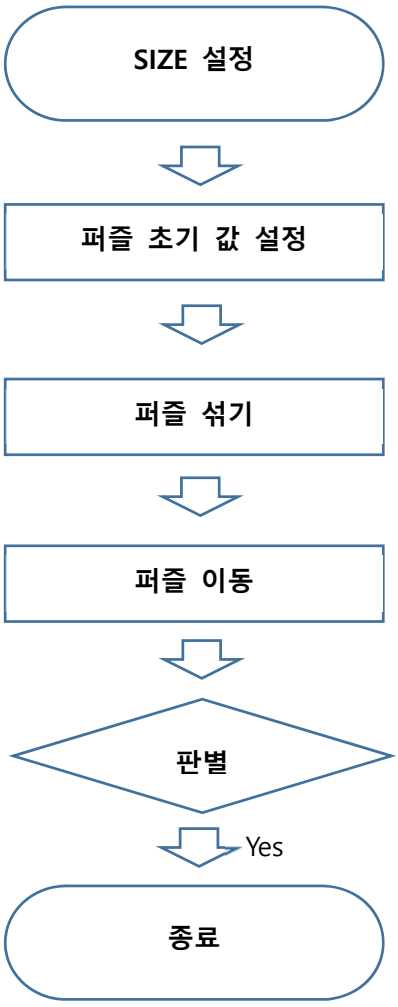
Report

Project 1 – Slide Game **12184407 이상운**

- **Game Method** : 1~Rows(가로)*Cols(세로)의 숫자들을 키보드의 입력으로부터 가로 줄부터 차례대로 오름차순으로 정렬을 한다.

- **Algorithm**



1. **3~7의 퍼즐 크기 숫자 입력**

   Ex) 3입력시 3X3 크기

2. **퍼즐의 숫자를 0~9까지 설정 (빈칸을 0으로 설정)**

3. **빈칸을 움직이며 카드를 섞음 (rand()이용)**

4. **키보드로 이동할 방향을 입력 받아 해당 퍼즐과 바꾼다.**

5. **[0][0]~[rows-1][cols-1]까지 숫자가 차례로 정렬 되었는지 확인한다.**

- **Constant**

```
#define MINP       3          - 퍼즐의 최소 크기 3으로 고정
#define MAXP       7          - 퍼즐의 최대 크기 7로 고정
#define OFFSET     5          - 프로그램 종료 시 문자열 좌표 값
#define KEY_ESC    0x1B       - ESC Key
#define KEY_8      '8'        - Up key
#define KEY_2      '2'        - Down key
#define KEY_4      '4'        - Left key
#define KEY_6      '6'        - Right key
#define KEY_W      'w'        - Up key
#define KEY_X      'x'        - Down key
#define KEY_A      'a'        - Left key
#define KEY_D      'd'        - Right key
```

- **A whole area variable**

```
int rows, cols;              퍼즐 크기(rows=가로줄, cols=세로줄)
int piece[MAXP][MAXP];       퍼즐 조각 (piece[7][7])
int blank_i, blank_j;        빈 조각의 위치
```

- **Main Function**

```
printf("Enter a size number (%d~%d) : ", MINP, MAXP);
scanf("%d", &rows);
if (rows < MINP || rows > MAXP)
      rows = MINP;
cols = rows;
```

① 사용자로부터 퍼즐 조각의 크기를 입력 받고 입력된 크기를 rows에 초기화 시킨다. 3~7외의 숫자가 입력되면 최소 크기(3)로 초기화 된다.

```
init();
```

② 배열 Piece[MAXP][MAXP]을 초기화 시키고 숫자를 섞는 함수

```
do
{
        move(getch());

} while (judge());
```

③ Do~while 구문으로 무조건 한번은 실행하게 한다. 사용자로부터 방향키를 입력 받아 빈칸의 퍼즐 조각을 움직이고 judge함수의 return 값으로 반복 문을 진행한다.

● Judge Function

```
for (i = 0; i < rows; i++)
{
        for (j = 0; j < cols; j++)
        {
                if (piece[i][j] != (j + 1) + i*rows)
                        return 1;

        }
}
return 0;
```

① Piece[0][0]부터 piece[cols-1][rows-1]까지 진행하면서 해당 배열의 숫자가 일치 하지 않을 경우 1을 return하면서 do~while 반복 문을 계속 실행한다. 만약 모든 숫자가 일치하여 if문의 조건이 만족하지 않으면 2중 for문을 빠져나와 0을 return하여 do~while 반복 문을 종료시킨다.

● Move Function

① 빈칸을 움직일 때, 반대로 생각해야 한다.

1	2	3
4	Blank	5
6	7	8

사용자로부터 왼쪽 이동의 키를 입력 받으면, Blank의 오른쪽에 있는 카드 '5'가 왼쪽으로 이동하여 Blank자리에 와야 한다. 따라서 카드의 이동은 Blank를 기준으로 하여 이동시켜야 한다. 또한 if문 조건식을 rows와 cols범위에 벗어나지 않도록 설정하여 가장자리에 위치한 숫자가 사각형 범위에 벗어나지 않게 한다.

```
switch (key)
{
case KEY_4:
case KEY_A:
        if (blank_i < rows - 1)
        {
                piece[blank_j][blank_i] = piece[blank_j][blank_i + 1];
                blank_i++;
        }
        break;
case KEY_6:
case KEY_D:
        if (blank_i > 0)
        {
        piece[blank_j][blank_i] = piece[blank_j][blank_i - 1];
        blank_i--;
        }
        break;
case KEY_2:
case KEY_X:
        if (blank_j > 0)
        {
```

```
            piece[blank_j][blank_i] = piece[blank_j - 1][blank_i];
            blank_j--;
        }
        break;
case KEY_8:
case KEY_W:
        if (blank_j < rows - 1)
        {
                piece[blank_j][blank_i] = piece[blank_j + 1][blank_i];
                blank_j++;
        }
        break;
case KEY_ESC:
        gotoxy(0, rows * 3 + 1 + OFFSET);
        printf("Exit!₩n₩n");
        exit(0);
        break;
}
```

② 사용자로부터 ESC키를 입력 받을 경우 exit(0)의 라이브러리 함수를 사용하여 프로그램을 종료시킨다.

```
piece[blank_j][blank_i] = rows*cols;
draw_piece(blank_i, blank_j);
```

③ 빈칸을 rows*cols 값으로 지정을 함으로 서, judge함수의 판단이 쉬워진다. Draw_piece 함수를 호출하여 카드의 그림을 그린다. 단, 빈칸의 위치 (blank_i, blank_j)를 알려주어 빈칸의 위치에는 그림이 없어야 한다.

```
for (i = 0; i < rows; i++)
{
        for (j = 0; j < cols; j++)
        {
                gotoxy(5 + j * 6, 5 + i * 3);
                if (piece[i][j] == rows*cols)
                        printf(" ");
                else
                        printf("%d ", piece[i][j]);
```

④ 빈칸의 숫자를 없애는 반복문으로서, 배열의 값을 분석하다가 rows*cols의 값을 발견하면 숫자를 없앤다.

● Init Function

```
blank_i = 0;
blank_j = 0;
for (i = 0; i < rows; i++)
{
        for (j = 0; j < cols; j++)
                piece[i][j] = j + +i*rows;
}
```

① 배열 piece[MAXP][MAXP]에 사용자로부터 입력 받은 퍼즐의 크기만큼 숫자를 초기화 시킨다.

```
srand((int)time(NULL));
```

```
for (i = 0; i < 10; i++)
{
        num = rand() % 4;
        if (num == 1)
                move('w');
        else if (num == 2)
                move("a");
        else if (num == 3)
                move("d");
        else if (num == 0)
                move("x");
}
```

② 시간에 따라 다르게 나타나는 난수 설정<srand((int)time(NULL))을 설정하고 num의 변수에 0~3까지의 난수를 무작위로 생성하여 그 난수에 해당하는 값이 지정한 move함수의 움직임에 넣어 빈칸이 돌아다니도록 설정한다. 카드는 컴퓨터로 섞지만, 사람이 섞는다고 생각하여야 한다.

```
draw_piece(blank_i, blank_j);
for (i = 0; i < rows; i++)
{
        for (j = 0; j < cols; j++)
        {

                gotoxy(5 + j * 6, 5 + i * 3);
                if (piece[i][j] == rows*cols)
                        printf(" ");
                else
                        printf("%d ", piece[i][j]);
        }
```

③ 퍼즐 그림과 섞인 조각들을 출력하며 move함수처럼 빈칸으로 지정된 위치에 숫자를 지우고 그림도 사라지게 한다.

● Draw_piece & Gotoxy Function

```
char piece_of_piece[3][7] = {
        { " ┌───┐ " },
        { " │   │ " },
        { " └───┘ " }
};
char blank[3][7] = {
        { "       " },
        { "       " },
        { "       " }
};
for (c = 0; c < rows; c++)
{
        for (b = 0; b < cols; b++)
        {
                for (a = 0; a < 3; a++)
                {
                        gotoxy(3 + b * 6, a + c * 3 + 4);
                        printf("%s", piece_of_piece[a]);
                }
        }
```

```
        }

        for (a = 0; a < 3; a++)
        {
                gotoxy(i * 6 + 3, j * 3 + 4 + a);
                puts(blank[a]);
        }

}
```

① 빈칸의 위치 정보 blank_i -> i / blank_j -> j를 move, init함수로 받아, 모든 위치에 그려진 퍼즐 조각들 중 빈칸의 위치에 있는 조각 그림을 지워준다.

```
void gotoxy(int x, int y)
{
        COORD Pos = { x, y };
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), Pos);
        }
```

② Gotoxy 함수 사용으로 커서의 위치를 이동 시켜, 사용자가 원하는 위치에 문자열 등을 프린트 할 수 있다.

Extra Section

1. 방향키 추가

```
#define UP          72
#define DOWN        80
#define LEFT        75
#define RIGHT       77
```

```
case KEY_4:       // left
case KEY_A:
case LEFT:
        if (blank_i < rows - 1)
        {
                piece[blank_j][blank_i] = piece[blank_j][blank_i + 1];
                blank_i++;
        }
        break;
```

- **태블릿 기반의 소형키보드로 게임을 하니, 기존의 이동 방식이 매우 불편하여 방향키로 편리하고 익숙한 방식을 사용하게 하였다.**

2. 게임 시작 시 경고음 추가

```
srand((int)time(NULL));
for (i = 0; i < Difficulty; i++) // Difficulty를 입력 받아 섞는 횟수 설정
{
        // 섞는 횟수가 클수록 카드가 섞이는게 보임
        num = rand() % 5;
        switch (num)
        {
        case 1:
                move('w');
                break;
        case 2:
                move('x');
                break;
        case 3:
                move('a');
                break;
        case 4:
                move('d');
        }

}
draw_piece(blank_i, blank_j);
        printf("₩a"); // 경고음 추가
```

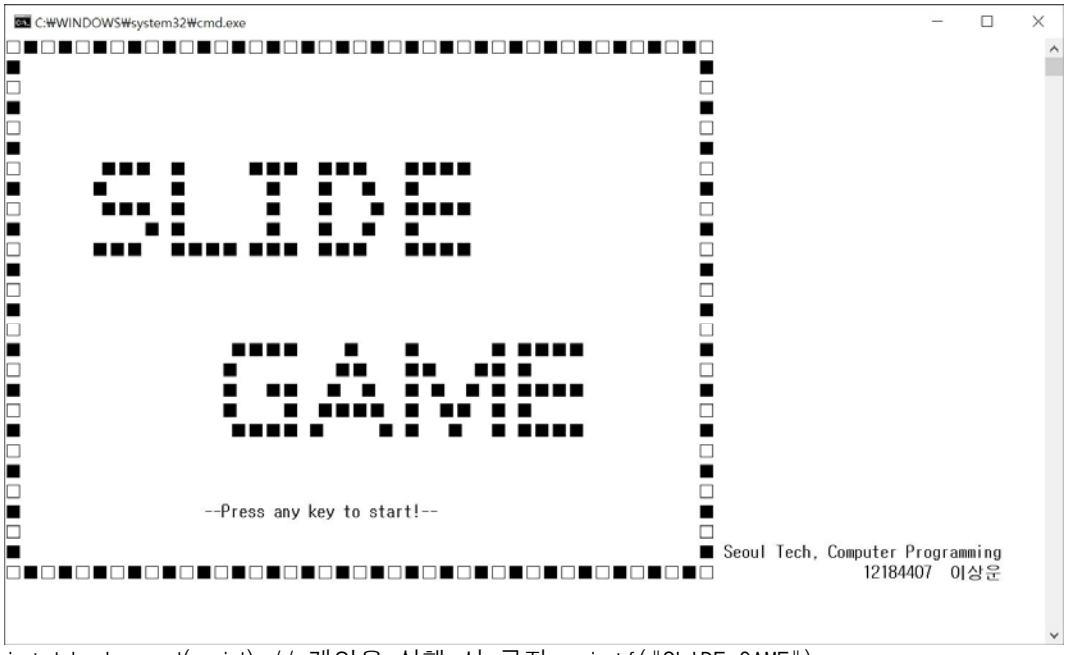
- **카드가 섞임과 동시에 게임이 시작 되므로 섞이고 나서 경고음을 추가하여 사용자에게 게임이 시작 되었다고 알린다.**

3. 메인 함수의 간소화

```
int main(void)
{
 start_page(); // 테두리 그림 함수
 black_word(); // 초기 시작 화면 글자 함수
 white_board(); // black_word함수의 글자 지우기 함수
 infor(); // 시작 시 설정 값 입력 및 조작 방법 printf 함수
 init();
 Cursor(); // 커서 깜빡임을 지우는 함수
 Score(); // 점수 매기기 함수
gotoxy(60, 23);
 printf("Done!");
 gotoxy(0, 28);
 return 0;
}
```

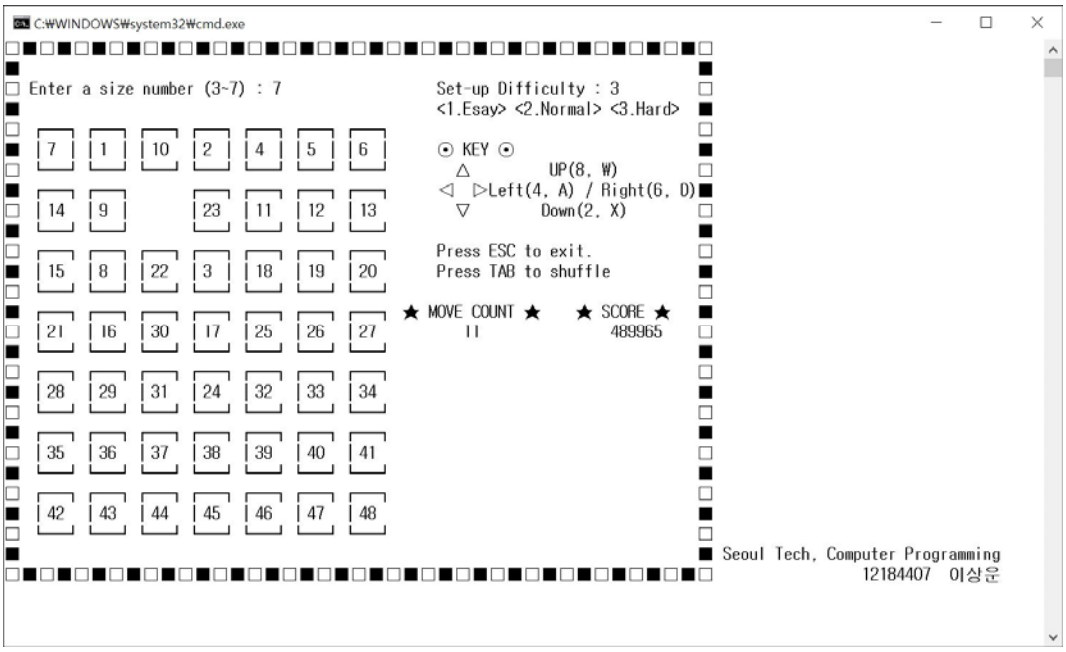
- **메인 함수가 복잡하고 지저분하게 보이는게 마음에 들지 않기 때문에 기존에 메인 함수에 있었던 do~while문, 퍼즐의 크기 입력 문 등을 다른 함수에 옮겼다.**

4. 배경화면 및 스타트 화면




```
int black_word(void) // 게임을 실행 시 글자 printf("SLIDE GAME")
{
        int i = 0;
        gotoxy(10, 6);
        printf(" ■■■  ■      ■■■  ■■■   ■■■■");
        gotoxy(10, 7);
        printf("■      ■        ■   ■   ■   ■");
        gotoxy(10, 8);
        printf(" ■■■  ■        ■   ■    ■  ■■■■");
        gotoxy(10, 9);
```

```
        printf("      ■ ■         ■    ■   ■   ■");
        gotoxy(10, 10);
        printf("■■■   ■■■■ ■■■  ■■■    ■■■■");

        gotoxy(25, 15);
        printf(" ■■■■     ■     ■       ■ ■■■■");
        gotoxy(25, 16);
        printf("■          ■■    ■■    ■■ ■");
        gotoxy(25, 17);
        printf("■   ■■   ■  ■   ■ ■  ■ ■ ■■■■");
        gotoxy(25, 18);
        printf("■     ■  ■■■■  ■  ■■  ■ ■");
        gotoxy(25, 19);
        printf(" ■■■■ ■      ■ ■   ■   ■ ■■■■");
        gotoxy(23, 23);
        printf("--Press any key to start!--");
        getch(i);
        if (i == '₩n') // 아무 키나 누를 시 다음 함수로 넘어감.
                return 0; }
```




```
void start_page(void) // Extra : 게임 테두리 그리기
{
        int i;
        printf("□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□
■□₩n");
        for (i = 0; i < 13; i++) // 좌측 세로줄 그리기
        {
                printf("■₩n");
                printf("□₩n");
        }
        for (i = 0; i < 13; i++) // 우측 세로줄 그리기
        {
        gotoxy(80, i * 2); // □이 짝수 세로줄에 printf된다 & ■이 홀수 세로줄에 printf된다.
                                printf("□");
```

```
                gotoxy(80, i * 2 + 1);
                printf("■");
        }
        gotoxy(0, 26);
        printf("□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□■□
■□₩n");
        gotoxy(82, 25);
        printf(" Seoul Tech, Computer Programming");
        gotoxy(82, 26);
        printf("                12184407  이상운");
        }
```

- **최대 크기인 7X7 퍼즐도 들어갈 테두리 크기 설정과 키 하나를 입력해야만 메인 페이지로 넘어 갈 수 있는 스타트 화면을 만들어, 기존 게임들과 비슷하게 표현하였다.**

5. 커서 숨기기

```
void Cursor(void)
{
        CONSOLE_CURSOR_INFO cur;
        GetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cur);
        cur.bVisible = 0;
        SetConsoleCursorInfo(GetStdHandle(STD_OUTPUT_HANDLE), &cur);
}
```

- **Gotoxy 함수와 비슷한 라이브러리 함수로 깜빡이는 커서를 보이지 않게 할 수 있다.**

6. 게임 난이도 설정

```
Set-up Difficulty : 3
<1.Esay> <2.Normal> <3.Hard>
```

```
        printf("<1.Esay> <2.Normal> <3.Hard>"); // 게임의 난이도 설정
        gotoxy(50, 2);
        printf("Set-up Difficulty : "); // Difficulty 전역 변수에 입력
        scanf("%d", &Difficulty);
        if (rows < MINP || rows > MAXP)
                rows = MINP;
        switch (Difficulty)
        {
        case 1: // 1 입력 시 10번 섞는다.
                Difficulty = 10;
                break;
        case 2: // 2 입력 시 50번 섞는다.
                Difficulty = 50;
                break;
        case 3: // 3 입력 시 100번 섞는다.
                Difficulty = 100;
                break;
        default: // 그 외의 숫자 입력 시 50번 섞는다.
                Difficulty = 50;
```

```
        }
for (i = 0; i < Difficulty; i++) // Difficulty를 입력 받아 섞는 횟수 설정
        {
                // 섞는 횟수가 클수록 카드가 섞이는게 보임
                num = rand() % 5;
                switch (num)
                {
                case 1:
                        move('w');
                        break;
                case 2:
                        move('x');
                        break;
                case 3:
                        move('a');
                        break;
                case 4:
                        move('d');
                }
```

- **난수를 이용하여 카드를 섞을 때, for문이 얼만 큼 실행 되는 횟수에 따라 섞이는 정도가 비례한다. 따라서 이를 이용하여 난이도를 설정 할 수 있다. 섞는 횟수가 많을수록 어려워지고 반대로 적게 섞을수록 쉬워진다.**
- **난이도를 3단계로 구분하여 1~3 외의 숫자를 입력 할 경우, 2(normal)의 난이도로 설정하였다.**
- **섞는 횟수가 많을수록 카드가 섞이는게 눈에 보이기 때문에 사용자로부터 더 큰 재미를 줄 수 있다.**

7. 이동 횟수 표시 & 점수 설정


```
void Score(void) // Extra : 게임 점수 제 설정, 이동 횟수 표시
{
        score = rows*cols*Difficulty * 100; // 초기 점수 값 (난이도와 퍼즐 개수에 따라 점수가 다름)
        gotoxy(46, 13);
        printf("★ MOVE COUNT ★     ★ SCORE ★₩n");

        do
        {
                gotoxy(46, 14);
                printf("       %d               %d", move_count/2, score);
                move(getch());
                ++move_count; // move_count 증가
                if (move_count >= 100) // move_count가 증가 할 때마다 score점수는 깎인다.
```

```
                score -= 10;       // 많이 증가할수록 점수는 더 많이 감소한다.
        else if (move_count >= 50)
                score -= 5;
        else if (move_count >= 10)
                score -= 2;
        else
                score -= 1;
        if (score < 0) // 점수가 음수일 시, 게임 종료
        {
                gotoxy(60, 22);
                printf("Game OVER!!!!");
                break; // judge함수와 상관 없이 do~while문 탈출
        }
} while (judge());
```

- **기존의 do~while 반복 문에 반복 문이 실행 될 때마다 move_count의 변수가 1식 증가하여 사용자가 방향키를 입력 할 때마다 증가 하는 것처럼 설정한다.**
- **점수(Score)를 적용하여 이동 횟수에 따라 점수가 깎이는 방식이다. 이동 횟수가 많아질수록 마이너스되는 점수도 커져 적은 이동 횟수로 게임을 클리어 해야한다.**
- **점수가 0미만으로 감소할 시, break;문으로 do~while 반복 문을 탈출하여 게임을 종료 시킨다.**
- **초기 점수 설정은 퍼즐 크기가 증가 할수록, 난이도가 클수록 클리어하기 어려움으로 초기 점수 또한 크게 설정된다.**

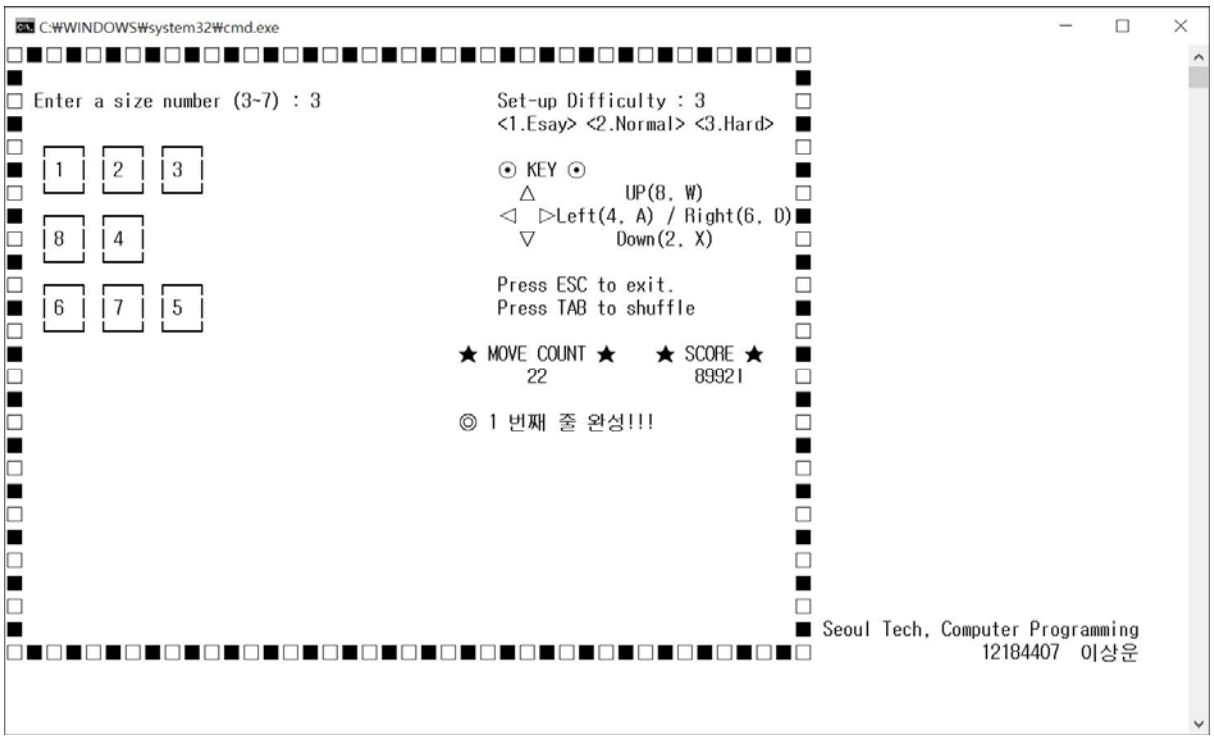
8. 게임 재 시작

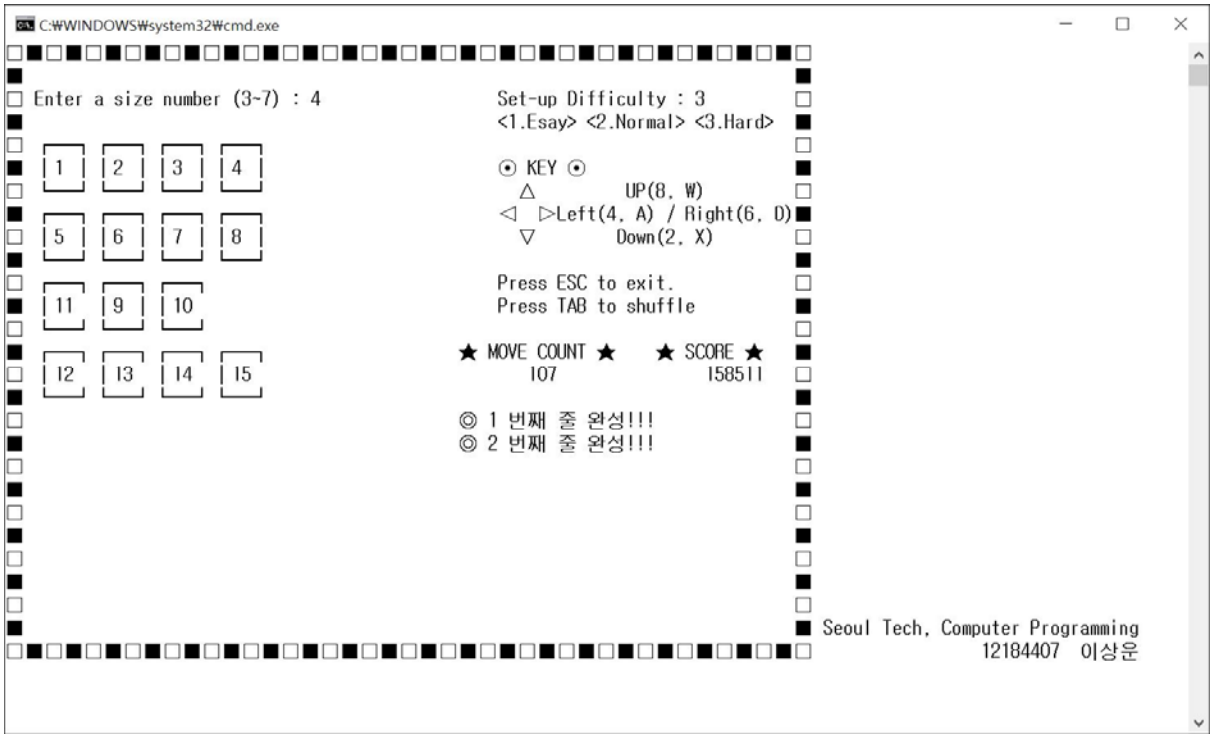
```
⊙ KEY ⊙
  △         UP(8, W)
◁  ▷Left(4, A) / Right(6, D)
  ▽        Down(2, X)

Press ESC to exit.
Press TAB to Restart
```

```
case'Wt':   // restart 기능 <TAB>
        gotoxy(46, 14);
        printf("                                ");
        init(); // 카드를 다시 섞는다.
        move_count = 0; // 움직인 횟수를 초기값으로 바꾼다.
        Score(); // 스코어 점수를 초기값으로 바꾼다.
        break;
```

- **'TAB' 키를 입력 받을 시, init함수로 다시 카드를 섞고 이동 횟수와 점수를 초기 값으로 바꾼다.**
-

9. 가로줄 판별로 인한 중간 점검






```
for (i = 0; i < rows; i++)
        {
                for (j = 0; j < cols; j++)
                {
                        temp++; // temp == 1부터 시작
                        if ((temp - 1) % rows == 0) // piece[j][i]가 제 위치에 자리 했을
                                                    때마다 temp 값이 1씩 증가됨
```

```
{
        if (temp <= rows) // 0번째 줄 printf방지
                continue;
        gotoxy(46, 16 + n);
        printf("◎ %d 번째 줄 완성!!!", temp / rows);
        n++;
}
if (piece[i][j] != (j + 1) + i*rows)
        return 1;
```

- **사용자로부터 방향키를 입력 받으면 judge함수의 2중 for문은 항상 실행이 된다. 이를 이용하여 변수 temp를 선언하여 가로줄이 차례로 맞춰 질 때 마다 1씩 증가하여 최종적으로 rows줄의 숫자가 모두 정렬이 되면 완성 되었다고 사용자에게 알려준다.**