1. 실습 시간에 작성한 프로그램이 예비보고서의 pseudocode와 달라진 점과 시간/공간 복잡도

(1) int CheckToMove(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX)

- 작성한 코드

```
int CheckToMove(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX)
{
    //블록이 명령에 따라 이동할 수 있는지 없는지 판단해 있다면 1, 없다면 0을 리턴한다.
    int i, j, x, y;

    for(i = 0; i < BLOCK_HEIGHT; i++) {
        for(j = 0; j < BLOCK_WIDTH; j++) {
            if(block[currentBlock][blockRotate][i][j] == 1) {
                y = blockY + i;
                x = blockX + j;

                //이동하려는 위치가 범위를 벗어난다면 0을 리턴한다.
                if (x < 0 || x >= WIDTH || y < 0 || y >= HEIGHT)
                    return 0;
                if (f[y][x] == 1)
                    return 0;
            }
        }
    }
    return 1;
}
```

pseudocode와 동일하게 작성했다. 시간 복잡도는 O(BLOCK_HEIGHT * BLOCK_WIDTH)이므로 O(1)이고, 공간 복잡도는 O(1)이다.

(2) void DrawChange(char field [HEIGHT][WIDTH],int command, int currentBlock, int blockRotate, int blockY, int blockX)

- 작성한 코드

```
void DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX){
    //명령에 따라 이동한/회전한 블록을 다시 그린다. (이전 블록을 지우고, 새 블록을 그린다.)
    int pre_blockRotate = blockRotate;
    int pre_blockY = blockY;
    int pre_blockX = blockX;

    int i, j;

    //Save the information of previous block.
    switch (command)
    {
        case KEY_UP:
            pre_blockRotate = (blockRotate + 3) % 4;
            break;
        case KEY_DOWN:
            pre_blockY -= 1;
            break;
        case KEY_LEFT:
            pre_blockX += 1;
            break;
        case KEY_RIGHT:
            pre_blockX -= 1;
    }

    //Erase the previous block.
    for (i = 0; i < BLOCK_HEIGHT; i++) {
        for (j = 0; j < BLOCK_WIDTH; j++) {
            if (block[currentBlock][pre_blockRotate][i][j] == 1) {
                //move cursor to (i + pre_blockY + 1, j + prev_blockX + 1) and print "."
                move(i + pre_blockY + 1, j + pre_blockX + 1);
                printw(".");
            }
        }
    }

    //Draw the new block.
    DrawBlock(blockY, blockX, currentBlock, blockRotate, ' ');
}
```

if, else if 문 대신 switch 문을 사용했다. 시간 복잡도는 O(BLOCK_HEIGHT * BLOCK_WIDTH)이므로 O(1)이고, 공간 복잡도는 O(1)이다.

(3) void BlockDown(int sig)

- 작성한 코드

```
void BlockDown(int sig){
    //현재 블록을 아래로 내릴 수 있다면 내린다.
    if (CheckToMove(field, nextBlock[0], blockRotate, blockY + 1, blockX)) {
        blockY++;
        DrawChange(field, KEY_DOWN, nextBlock[0], blockRotate, blockY, blockX);
    }

    //현재 블록을 아래로 내릴 수 없는 경우.
    else {
        //블록이 시작부터 내릴 수 없다면, 필드에 블록을 놓을 공간이 없다는 의미이므로 게임오버이다.
        if (blockY == -1) gameOver = 1;

        //블록을 필드에 쌓는다.
        AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX);

        //빈 칸 없는 줄이 있는지 확인해서 삭제하고, 점수를 계산한다.
        score += DeleteLine(field);

        //현재 블록과 다음 블록 설정
        nextBlock[0] = nextBlock[1];
        nextBlock[1] = rand() % 7;

        //블록의 위치 초기화
        blockY = -1;
        blockX = WIDTH/2 - 2;
        blockRotate = 0;

        //다음 블록을 디스플레이한다.
        DrawNextBlock(nextBlock);

        //점수를 디스플레이한다.
```

```
            PrintScore(score);

            DrawField();
        }
        //블록이 다 떨어졌으므로 0으로 초기화한다.
        timed_out = 0;
}
```

블록이 내려갈 수 있는지 확인하는 조건문에서 CheckToMove() 함수를 사용했다. CheckToMove(), DrawChange(), AddBlockToField(), DeleteLine(), DrawNextBlock() 함수 모두 O(1)이므로 시간 복잡도는 O(1)이고, 공간 복잡도는 O(1)이다.

(4) void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX)

- 작성한 코드

```
void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX){
    //해당 좌표에 맞게 필드에 현재 블록을 쌓는다.

    int i, j, x, y;

    for (i = 0; i < BLOCK_HEIGHT; i++) {
        for (j = 0; j < BLOCK_WIDTH; j++) {
            //블록의 모양대로 추가된 영역의 필드 값을 바꾼다.
            if (block[currentBlock][blockRotate][i][j] == 1) {
                y = blockY + i;
                x = blockX + j;
                f[y][x] = 1;
            }
        }
    }
}
```

pseudocode와 동일하게 작성했다. 시간 복잡도는 O(BLOCK_HEIGHT, BLOCK_WIDTH)이므로 O(1)이고, 공간 복잡도는 O(1)이다.

(5) int DeleteLine(char f[HEIGHT][WIDTH])

- 작성한 코드

```
int DeleteLine(char f[HEIGHT][WIDTH]){
    //빈 칸이 없는 줄을 찾아서 지우고, 지워진 줄 개수에 따라 점수를 계산해 리턴한다.

    //1. 필드를 탐색하여, 꽉 찬 구간이 있는지 탐색한다.
    //2. 꽉 찬 구간이 있으면 해당 구간을 지운다. 즉, 해당 구간으로 필드값을 한칸씩 내린다.
    int i, j, x, y;
    int count = 0;
    bool filled_flag;

    //필드를 위에서부터 아래로 탐색한다.
    for (i = 0; i < HEIGHT; i++) {
        //완전히 1로 채워진 줄을 찾는다.
        for (j = 0; j < WIDTH; j++) {
            if (!f[i][j]) {
                filled_flag = false;
                break;
            }
            else
                filled_flag = true;
        }

        //완전히 1로 채워진 줄인 경우
        if (filled_flag) {
            //점수 계산을 위해 지운 줄 개수를 센다.
            count++;

            //지운 줄 바로 위부터 가장 위까지, 필드를 내려준다.
            for (y = i; y >= 0; y--) {
                for (x = 0; x < WIDTH; x++) {
                    f[y][x] = f[y-1][x];
                }
            }
            for (x = 0; x < WIDTH; x++) { //가장 윗 줄은 새로 생겼으므로 0으로 셋팅한다.
                f[0][x] = 0;
            }
            //한 줄 지워주고 나머지 필드를 내려줬으므로 i--
            i--;
```

```
        }
    }

    return count * count * 100;
}
```

pseudocode와 동일하게 C언어 문법으로 작성했다. 시간 복잡도는 O(HEIGHT * WIDTH)이므로 O(1)이고, 공간 복잡도는 O(1)이다.

2. 테트리스 프로젝트 1주차 숙제 문제 해결을 위한 pseudocode와 시간/공간 복잡도

<tetris.h>

```
…
#define BLOCK_NUM 3
…
```

<tetris.c>

(1) 그림자 기능

```
void DrawBlock(y, x, blockID, blockRotate, tile)
    for (i = 0 to 3)
        for (j = 0 to 3)
            if (block[blockID][blockRotate][i][j] == 1 && i+y ≥ 0)
                move cursor to (i+y+1, j+x+1)
                print color-reversed tile
    move cursor to (HEIGHT, WIDTH+10)
```

시간 복잡도: O(4 * 4)이므로 O(1), 공간 복잡도:O(1)

```
void DrawShadow(y, x, blockID, blockRotate)
    int i = 0
    while (The block can move further down)
        i++
    i--
    DrawBlock(y + i, x, blockID, blocRotate, '/')
```

시간 복잡도: while 문(최대 bound는 HEIGHT)과 DrawBlock()에 의존하여 O(1), 공간 복잡도: O(1)

```
void DrawBlockWithFeatures(y, x, blockID, blockRotate)
    DrawBlock(y, x, blockID, blockRotate, ' ')
    DrawShadow(y, x, blockID, blockRotate)
```

시간 복잡도: DrawBolck(), DrawShadow()에 의존하여 O(1), 공간 복잡도: O(1)

(2) 미리 보여주는 블록을 2개로 변경

```
void InitTetris()
    for (j = 0 to HEIGHT-1)
        for (i = 0 to WIDTH-1)
            field[j][i] = 0

    nextBlock[0] = random integer in 0~6
    nextBlock[1] = random integer in 0~6
    nextBlock[2] = random integer in 0~6
    blockRotate = 0
    blockY=-1
    blockX=WIDTH/2-2
    score=0
    gameOver=0
    timed_out=0

    DrawOutline()
    DrawField()
    DrawBlockWithFeatures(blockY,blockX,nextBlock[0],blockRotate)
    DrawNextBlock(nextBlock)
    PrintScore(score)
```

시간 복잡도: HEIGHT * WIDTH에 및 호출 함수들에 의존하여 O(1), 공간 복잡도: O(1)

```
void DrawNextBlock(nextBlock)
    for( i = 0 to 3)
        move cursor to (4+i, WIDTH+13)
        for( j = 0 to 3)
            if( block[nextBlock[1]][0][i][j] == 1 ){
                print color-reversed blank
            else
                print blank
```

```
    for( i = 0 to 3)
        move cursor to (10+i, WIDTH+13)
        for( j = 0 to 3)
            if( block[nextBlock[2]][0][i][j] == 1 ){
                print color-reversed blank
            else
                print blank
```

시간 복잡도: O(4 * 4)이므로 O(1), 공간 복잡도: O(1)

```
void BlockDown(sig)
        if (block can go down)
                blockY++
                DrawChange(field, KEY_DOWN, nextBlock[0], blockRotate, blockY, blockX)
        else
                if (blockY == -1) gameOver = 1
                score += AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX)
                score += DeleteLine(field)
                nextBlock[0] = nextBlock[1]
                nextBlock[1] = nextBlock[2]
                nextBlock[2] = rand() % 7
                blockY = -1
                blockX = WIDTH/2 – 2
                blockRotate = 0
                DrawNextBlock(nextBlock)
                PrintScore(score)
                DrawField()
        timed_out = 0
```

시간 복잡도: 호출하는 함수들에 의존하여 O(1), 공간 복잡도: O(1)

(3) 닿은 면적에 따라 score 증가시키기

```
void AddBlockToField(f[HEIGHT][WIDTH], currentBlock, blockRotate, blockY, blockX)
        Set touched = 0
        for (i = 0 to BLOCK_HEIGHT-1)
                for (j = 0 to BLOCK_WIDTH-1)
                        if (block[currentBlock][blockRotate][i][j] == 1)
                                y = blockY + i
                                x = blockX + j
                                f[y][x] = 1
                                if (block touched the ground || block touched stacked blocks)
                                        touched++
        return touched * 10
```

시간 복잡도: O(BLOCK_HEIGHT * BLOCK_WIDTH)이므로 O(1), 공간 복잡도: O(1)

```
void BlockDown(sig)
        if (block can go down)
                blockY++
                DrawChange(field, KEY_DOWN, nextBlock[0], blockRotate, blockY, blockX)
        else
                if (blockY == -1) gameOver = 1
                score += AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX)
                score += DeleteLine(field)
                nextBlock[0] = nextBlock[1]
                nextBlock[1] = nextBlock[2]
                nextBlock[2] = rand() % 7
                blockY = -1
                blockX = WIDTH/2 – 2
                blockRotate = 0
                DrawNextBlock(nextBlock)
                PrintScore(score)
                DrawField()
        timed_out = 0
```

시간 복잡도: O(1), 공간 복잡도: O(1)

((2)에서 작성한 BlockDown() pseudocode와 동일하다.)