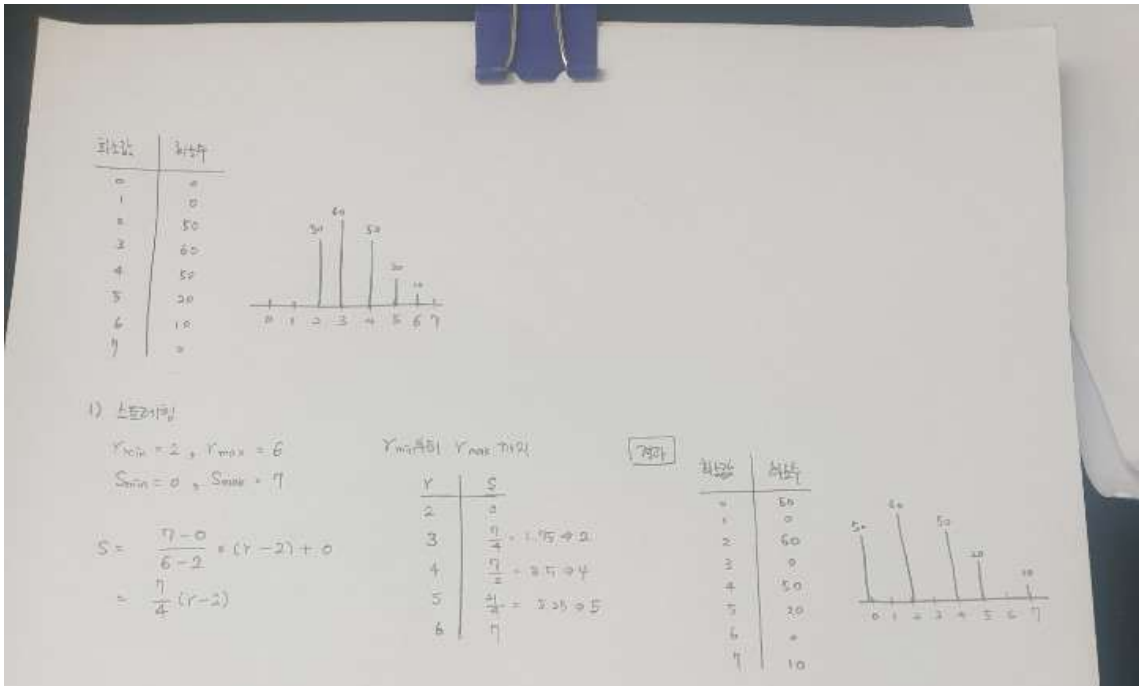
HW 1 풀이

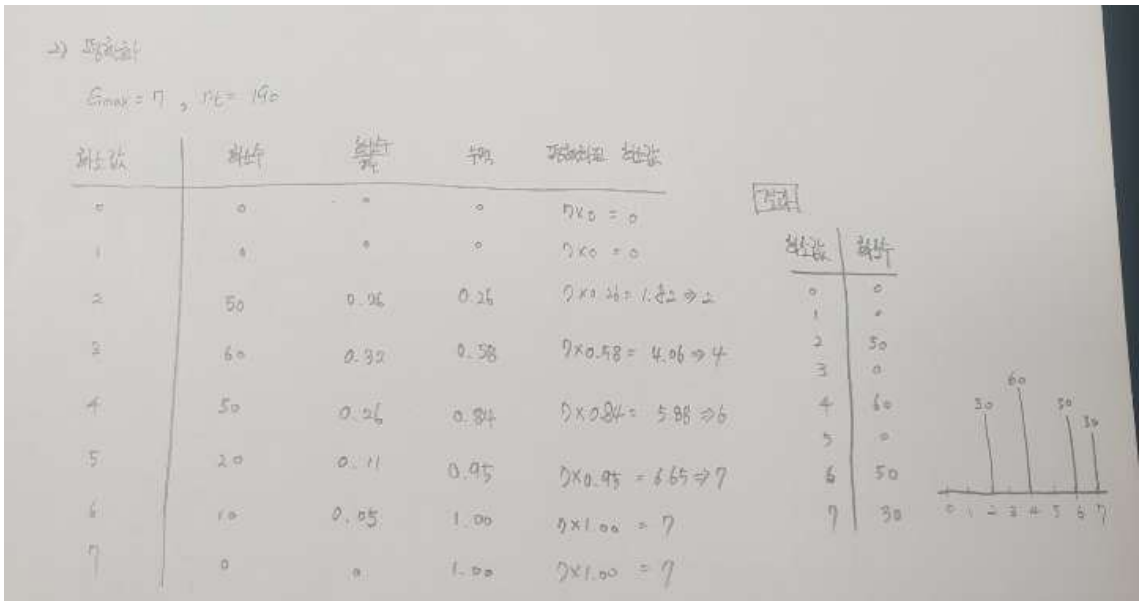
1) 히스토그램 스트레칭

위의 문제 같은 경우, 소수점으로 계산된 값을 정수값으로 변환하는 과정에서 다양한 학생들의 풀이가 있었습니다. **교수님께서는 상관없다고 하셨지만**, 이론 문제로 나온다면 채점을 위해 통일 해야 하기 때문에, 소수점을 정수값으로 바꾸는 과정은 '반올림'으로 통일하겠습니다.



2) 히스토그램 평활화

위의 문제 같은 경우, 학생들은 Gmax를 강의자료에 있는 255값을 사용하거나 임의로 설정하였습니다. 강의자료에는 실제 이미지의 한 부분을 가져와서 255로 설정하였던 것이고, **별도의 언급이 없었기 때문에** Gmax 값은 '표의 최대 밝기값'으로 통일하겠습니다.



HW 2 풀이

많은 학생들이 reduce 함수를 사용하지 못하였다고 생각합니다.
reduce 함수는 (InputArray src, OutputArray dst, int dim, int rtype)가 순서대로 들어가며, src는 입력, dst는 reduce한 출력, **dim**은 0일 경우 **한 행**으로 감축(y방향 projection), 1일 경우, **한 열**로 감축(x방향 projection)입니다. **rtype** 같은 경우 감축 연산 방법을 결정하는데, [REDUCE_SUM, REDUCE_AVG, REDUCE_MAX, REDUCE_MIN] 이 입력됩니다. 감축 연산 방법은 직접 해보시면서 비교하시기 바랍니다.

```
int main()
{
    Mat image = imread("equalize_test.jpg", 0);
    CV_Assert(!image.empty());

    cv::Mat projection_x, projection_y;
    cv::reduce(image, projection_x, 1, cv::REDUCE_AVG);
    cv::reduce(image, projection_y, 0, cv::REDUCE_AVG);

    cv::Mat draw_x, draw_y;
    draw_histo_user(projection_x, draw_x);
    draw_histo_user(projection_y, draw_y);

    imshow("image", image);
    imshow("projection_x", draw_x);
    imshow("projection_y", draw_y);
    waitKey();
}
```

```
void draw_histo_user(Mat hist, Mat& hist_img)
{
    cv::Size size;
    if (hist.cols == 1)
    {
        // x방향 projection
        size = cv::Size(256, hist.rows);
        hist_img = Mat(size, CV_8U, Scalar(255));       // 그래프 행렬

        for (int i = 0; i < hist.rows; i++)
        {
            float start_y = i;                  // 막대 사각형 시작 y 좌표
            float end_y = (i + 1);              // 막대 사각형 종료 y 좌표
            Point2f pt1(0, start_y);
            Point2f pt2(hist.at<uchar>(i), end_y);

            if (pt2.x > 0)
            {
                rectangle(hist_img, pt1, pt2, Scalar(0), -1);   // 막대 사각형 그리기
            }
        }
    }
    else
    {
        // y방향 projection
        size = cv::Size(hist.cols, 256);
        hist_img = Mat(size, CV_8U, Scalar(255));       // 그래프 행렬

        for (int i = 0; i < hist.cols; i++)
        {
            float start_x = i;                  // 막대 사각형 시작 x 좌표
            float end_x = (i + 1);              // 막대 사각형 종료 x 좌표
            Point2f pt1(start_x, 0);
            Point2f pt2(end_x, hist.at<uchar>(i));

            if (pt2.y > 0)
            {
                rectangle(hist_img, pt1, pt2, Scalar(0), -1);   // 막대 사각형 그리기
            }
        }

        flip(hist_img, hist_img, 0);                            // x축 기준 영상 뒤집기
    }
}
```