ACM ICPC2014 国内予選　問題 B 連鎖消滅パズル

【考察】

- ３個以上水平に同じ数字の石が並んでいれば、これらは同時に消滅し空きを埋めるように落下する。消滅する石がなくなるまで、この操作を繰り返す。消滅した石の数字の合計が獲得したスコアであり、これを求める。
- H 行５列の盤面の情報は 2 次元配列に記憶する。
- ５列しかないので、３個以上水平に同じ数字が並ぶのは、各行に対して高々一通りしかないので、これを各行ごとに求め、それらの石を消去（石の数字を 0 にする）するとともに、消滅した石の数字の合計をスコアに加える。
- 石の落下は、各列ごとに、バブルソートの要領で、「0」の石を上に上げる操作を繰り返す。

【プログラム例】

```
// ACM-ICPC2014 Domestic Problem B 連鎖消滅パズル
//   http://icpc.iisf.or.jp/past-icpc/domestic2014/#section_B
//   Filename = pb.c
//   Compile: cc pb.c
//   Execution: ./a.out < B0 > B0.out
//   Check:          diff B0.ans B0.out
#include <stdio.h>
#define MAXH 10
int map[MAXH][5];           // 盤面を記憶する 2 次元配列
int h;                      // 盤面の行数

int check(void)             // 消滅する石とそのポイントを求める
{
  int row, col, len, i;
  int score = 0;

  for(row=0; row<h; row++){           // 各行に対して
    for(col = 0; col <= 2; col++){    // 各開始列に対して
      for(len=1; col+len<=4; len++){  // 隣が同じ数字かを順にチェック
        if(map[row][col] != map[row][col+len]) break;    // 異なるので break
      }
      if(len<=2) continue;  // 同じ数字が並ぶ長さが 2 以下なので次の開始列へ
      score += map[row][col]*len;               // この行の消滅ポイントを加える
      for(i=0; i<len; i++)
        map[row][col+i] = 0;                    // 石を消滅させる
    }
  }
```

```
    return score;
}

void down(void)                 // 消滅した石の上の石を下へ落とす
{
    int col, row, min;
    for(col=0; col<5; col++){               // 列 col に対してバブルソートの要領で
                                            // 消滅した石を上に上げていく
        for(min=1; min<h; min++){           // min 行まで石を落とせるかをチェック
            for(row=h-1; row>=min; row--){          // 一番下の行から min 行まで
                if(map[row][col] == 0){             // ここの石が消滅していたら
                    map[row][col] = map[row-1][col];    // 一つ上の石をここに落とす
                    map[row-1][col] = 0;                // 一つ上の石は消滅
                }
            }
        }
    }
}

int main()
{
    int i,j;
    int score,cs;
    while(1){
        scanf("%d", &h);            // 盤面の行数 h を入力
        if(h==0) break;             // h=0  なら終了
        for(i=0; i<h; i++)          // 盤面データの読み込み
            for(j=0; j<5; j++)
                scanf("%d", &map[i][j]);
        score = 0;                  // 消滅ポイントの合計を 0 に初期化
        while(1){                   // 消滅する石がなくなるまで繰り返す
            cs = check();           // 現在の盤に対する消滅ポイントを求める
            if(cs == 0){            // 消滅する石がないので
                printf("%d¥n", score);          // これまでの消滅ポイント合計が答え
                break;
            }
            score += cs;            // 今回の消滅ポイントを合計に加える
            down();                 // 消滅した石の上の石を下へ落とす
        }
    }
}
```