ACM ICPC2013 国内予選　問題 B　ICPC の順位付け

- ICPC の順位付けを行う問題。順位は、
  1. より多くの問題に正解したチームが上位
  2. 正解問題数が同じ場合は、所要時間合計の少ないチームが上位
  3. 正解問題数が同じで所要時間合計も同じチームは同順位
- 所要時間合計 ＝ 正解した問題の所要時間の和
- 正解した問題の所要時間
  ＝ 正解時の経過時間 ＋20 × その問題の不正解数
- 出力形式：　上位から順に「コンマ」区切りでチーム番号を出力
  ただし、同順位のチーム間の区切りは「イコール」でチーム番号の大きい物から出力する
- Algorithm
  1. 提出記録に基づき、対応するチームの正解数、所要時間合計、対応する問題の不正解数を更新する。すなわち、
     (ア) 提出記録が不正解の場合：　対応する問題の不正解数を増やす
     (イ) 提出記録が正解の場合：　提出記録の経過時間 ＋20×その問題の不正解数（ペナルティ）を所要時間合計に加算
  2. チームデータを ICPC 順位でソート (ICPC 順位が同順位の場合はチーム番号の降順)
  3. 一つ前のチームと同順位なら「イコール」、異なる順位なら「コンマ」を出力後、チーム番号を出力

【プログラム例】

```
// ACM ICPC2013 Domestic Problem B
// http://sparth.u-aizu.ac.jp/icpc2013/d_problems.php?lang=jp#section_B
// Filename:        pb.c
// Compile:         cc pb.c
// Execution:       ./a.out < B0 > B0.result
// Check:   diff B0.result B0.ans
// Algorithm:
//          提出記録に基づき各チームのデータ（正解数、所要時間合計、不正解数）を更新
//          正解の場合は正解数を増やし、不正解数のペナルティを考慮した所要時間を加算
//          チームデータをソートし、出力形式を実現

#include <stdio.h>
#include <stdlib.h>
#define MAX_T       50          // チーム数の最大値
```

```
#define MAX_P       10          // 問題数の最大値

struct team_t {
   int id;                      // チーム id
   int ns;                      // 正解数
   int st;                      // 所要時間合計
   int fail[MAX_P];             // 各問題の不正解数
} team[MAX_T];                  // チームデータ

int compar(const void *p1, const void *p2)          // Sorting のための比較関数
{
   struct team_t *t1 = (struct team_t *)p1;
   struct team_t *t2 = (struct team_t *)p2;
   if(t1->ns > t2->ns) return -1; // 正解数が多い方が上位(前)
   else if(t1->ns < t2->ns) return 1;       // 正解数が少ない方が下位(後ろ)
   else {                       // 正解数が同じ場合は
      if(t1->st < t2->st) return -1;        // 所要時間合計が少ない方が上位(前)
      else if(t1->st > t2->st) return 1;    // 所要時間合計が多い方が下位(後ろ)
      else {                    // 所要時間合計も同じ場合は
         if(t1->id > t2->id) return -1;     // チーム番号の大きい方が前
         else return 1;                     // チーム番号の小さい方が後ろ
      }
   }
}

int main()
{
   int i,k;
   int M, T, P, R;
   int m, t, p, j;
   while(1){
      scanf("%d %d %d %d", &M, &T, &P, &R);
      if(M==0 && T==0 && P==0 && R==0) break;  // すべて 0 なら終了
      for(i=0; i<T; i++){           // 各チームの
         team[i].ns = team[i].st = 0;       // 正解数と所要時間合計を 0 に初期化
         team[i].id = i+1;                  // チーム id をセット
         for(j=0; j<P; j++)
              team[i].fail[j] = 0;          // 各問題の不正解数を 0 に初期化
      }
      for(i=0; i<R; i++){
         scanf("%d %d %d %d", &m, &t, &p, &j);
         if(j) team[t-1].fail[p-1]++;                   // 不正解なら不正解数を増やす
         else {                                        // 正解なら
              team[t-1].ns++;                          // チームの正解数を増やす
              team[t-1].st += m + 20*team[t-1].fail[p-1];       // 不正解のペナルティを
                                                                //考慮して所要時間を追加
         }
      }
      qsort(team, T, sizeof(struct team_t), compar);    // チームデータをソート
      for(i=0; i<T; i++){
         int pns, pst;   // 一つ前にプリントしたチームの正解数と所要時間合計
         if(i==0)
```

```
            printf("%d",team[i].id);            // 最初のチーム名をプリント
        else {
            if(team[i].ns == pns && team[i].st == pst)
                printf("=%d", team[i].id);      // 前のチームと同順位
            else
                printf(",%d", team[i].id);      // 前のチームより後の順位
        }
        // このチームの正解数と所要時間合計を一つ前のものとして保存
        pns = team[i].ns;
        pst = team[i].st;
    }
    printf("¥n");
  }
}
```