ACM ICPC 2015 国内予選　問題 A 入学試験

【考察】

- 「条件を満たす n が存在することが保証される」と記載されているので、仮に同得点者の間で線引きしてもギャップは 0 なので、この線引きが採用されることは無い。即ち、同得点者がいることを考慮したプログラムにする必要はない。
- 受験者の得点を $P_1, P_2, \cdots\cdots, P_n, P_{n+1}, \cdots\cdots, P_m$ とすると、点数は降順に並んでいるので、合格者数を n とすると、P_n が合格最低点であり P_{n+1} が不合格者の最高点となる。したがって、ギャップは $P_n - P_{n+1}$ となる。
- $n_{min} <= n <= n_{max}$ なる n の中で、ギャップが最大となるものを求め、最大ギャップの中での n の最大値を求めれば良い。
- 得点 P_i を配列 p[i – 1]に格納して計算する方法（プログラム pa1.c 参照）：
  - 合格者数が n の場合のギャップは p[n] – p[n-1]となる。
  - $n = n_{min}$ から n_{max} まで順にギャップを計算し、これまでに得られたギャップ以上のギャップになった時のギャップ値と合格者数を記憶しておけば良い。最終的に記憶されている合格者数が求める値である。
- 配列を用いない用法（プログラム pa2.c 参照）
  - n+1 人目に読み込んだ得点を score とし、一つ手前で読み込んだ得点（n 人目の得点）を prev とすると、prev – score が合格者数 n の時のギャップである。
  - 得点を順番に読み込み、n_{min}+1 回目から n_{max}+1 回目までの間で、これまでに得られたギャップ以上のギャップになった時のギャップ値と合格者数を記憶しておけば良い。最終的に記憶されている合格者数が求める値である。

【プログラム例 pa1.c　（配列を使用）】

```
// ACM-ICPC 2015 国内予選 Problem A 入学試験
// http://icpc.iisf.or.jp/past-icpc/domestic2015/contest/all_ja.html#section_A
//   Filename:      pa1.c
//   Compile:       gcc pa1.c
//   Execution:     ./a.out < A0 > A0.result
//   Check:         diff A0.ans A0.result
//   Algorithm:      受験者の得点を配列に格納してから、
//                  nmin <= n <= nmax なる合格者数 n に対し順にギャップを求め
//                  ギャップが最大の時の合格者数の最大値を求める。

#include <stdio.h>
#define MAXM 200            // 受験者数の最大値
int m;                      // 受験者数
int nmin;                   // 最小合格者数
int nmax;                   // 最大合格者数
int p[MAXM];                // 各受験者の得点（降順）p[i-1] = Pi

int main(void)
{
  for(;;){
    int i, n;               // n = 合格者数
    int gap;                // これまでに求まっているギャップ
    int nans;               // これまでに求まっている合格者数
    scanf("%d %d %d", &m, &nmin, &nmax); // 受験者数、最小/最大合格者数の入力
    if(m==0 && nmin==0 && nmax==0) break;          // 全て 0 なら終了
    for(i=0; i<m; i++)
      scanf("%d", &p[i]);               // 点数を読み込む（降順にソート済み）
    gap = 0;
    // nmin <= n <= nmax なる各 n に対して順にギャップを計算
    // 合格者数が n の時のギャップは、p[n-1]-p[n] となる。
    for(n=nmin; n<=nmax; n++){
      if(p[n-1]-p[n] >= gap){// これまでのギャップ以上なら
        gap = p[n-1]-p[n];  // そのギャップを記憶し
        nans=n;                         // n を暫定合格者数とする
      }
    }
    printf("%d¥n", nans);               // 答えの出力
  }
  return 0;
}
```

【プログラム例 pa2.c　（配列未使用）】

```
// ACM-ICPC 2015 国内予選 Problem A 入学試験
// http://icpc.iisf.or.jp/past-icpc/domestic2015/contest/all_ja.html#section_A
//   Filename:      pa2.c
//   Compile:       gcc pa2.c
//   Execution:     ./a.out < A0 > A0.result
//   Check:         diff A0.ans A0.result
//   Algorithm:     受験生の得点を読み込みながらギャップを求め
//                  ギャップが最大の時の合格者数の最大値を求める。
//                  受験生の得点を覚える配列は不要

#include <stdio.h>

int m;                          // 受験者数
int nmin;                       // 最小合格者数
int nmax;                       // 最大合格者数

int main(void)
{
  int prev;                     // 合格最低点
  int score;                    // 不合格最高点
  for(;;){
    int n;                      // 合格者数
    int gap;                    // これまでに求まっているギャップ
    int nans;                   // これまでに求まっている合格者数
    scanf("%d %d %d", &m, &nmin, &nmax); // 受験者数、最小/最大合格者数の入力
    if(m==0 && nmin==0 && nmax==0) break;          // 全て 0 なら終了
    gap = 0;
    scanf("%d", &score);        // 一人目の得点
    for(n=1; n<m; n++) {        // n = 合格者数
      prev = score;                     // prev = n 人目の合格者の得点
      scanf("%d", &score);  // score = 最初の不合格者の得点
      if(n < nmin || n > nmax) continue;
      // nmin <= n <= nmax なる n がチェック対象
      if(prev - score >= gap){          // 合格者数が n の時のギャップは prev -
score
        // このギャップがこれまでに求まった最大ギャップ以上なら、
        // このギャップ値と合格者数を保存
        gap = prev-score;
        nans = n;
      }
    }
    printf("%d¥n", nans);   // 答えの出力
  }
  return 0;
}
```