Solutions for Exam #4 (1/6/2016)

Professors of PRO1

1 General comments

In several of the solutions below, we have omitted the parts of the C++ code that were already given in the corresponding problem statements, and have thus concentrated in the fragments of code that had to be designed to solve the problems.

2 Exercises

2.1 X62240: How many constant columns?

```
// ...

// return true iff j-th column of matrix M is constant
bool is_constant_column(const Matrix& M, int j) {
  const int numfil = M.size();
  for (int i = 0; i < numfil - 1; ++i)
    if (M[i][j] != M[i+1][j]) return false;
  return true;
}

int how_many_constant_columns(const Matrix& M)
{
  int const_col = 0;
  const int numcol = M[0].size();
  for (int j = 0; j < numcol; ++j)
    if (is_constant_column(M, j))
      ++const_col;
  return const_col;
}
```

2.2 X82451: Points inside rectangles

```
// ...
// Returns "inside", "border" or "outside" depending on whether
```

```
// p is inside, at the border, or outside of r.
string containtment(Point p,Rectangle r)
{
  if (r.x1 < p.x and p.x < r.x2 and
      r.y1 < p.y and p.y < r.y2)
    return "inside";
  if (r.x1 <= p.x and p.x <= r.x2 and
      r.y1 <= p.y and p.y <= r.y2)
    return "border";

  return "outside";
}

int main()
{
  int n;
  cin >> n;
  while (n > 0) {
    Point p = read_point();
    Rectangle r = read_rectangle();
    cout << containtment(p,r) << endl;
    --n;
  }
}
```

2.3 X61415: Sort rectangles

```
// ...

// read n rectangles from the standard input
vector<Rectangle> read_rectangles(int n) {
  vector<Rectangle> v(n);
  for (int i = 0; i < n; ++i)
    cin >> v[i].xmin >> v[i].ymin
        >> v[i].xmax >> v[i].ymax;
  return v;
}

// print the rectangles, end each rectangle followed by a line break.
void print_rectangles(const vector<Rectangle>& v) {
  for (int i = 0; i < int(v.size()); ++i)
    cout << v[i].xmin << ' ' << v[i].ymin
         << ' ' << v[i].xmax << ' ' << v[i].ymax << endl;
}

// Pre: R.xmin<R.xmax and R.ymin<R.ymax.
```

```
// returns the area of R.
int area(const Rectangle& R)
{
  return (R.xmax-R.xmin)*(R.ymax-R.ymin);
}

// returns true if and only if R1 is smaller than R2 (see the statement for a
bool smaller(const Rectangle& R1, const Rectangle& R2) {
  if (area(R1) < area(R2)) return true;
  if (area(R1) == area(R2) and R1.xmin < R2.xmin) return true;
  if (area(R1) == area(R2) and R1.xmin == R2.xmin
      and R1.ymin < R2.ymin) return true;
  if (area(R1) == area(R2) and R1.xmin == R2.xmin
      and R1.ymin == R2.ymin and R1.xmax < R2.xmax) return true;
  return false;
}

int main() {
  int n;
  cin >> n;
  vector<Rectangle> v = read_rectangles(n);

  sort(v.begin(), v.end(), smaller);

  print_rectangles(v);
}
```

2.4 X41129: Common elements

```
// Pre: A[low] <= x < A[high], A[-1] = -infinity, A[n]=+infinity,
// -1 <= low < high <= n

bool occurs(const vector<int>& A, int low, int high, int x) {
  if (low + 1 == high)
    return low != -1 and A[low] == x;
  int m = (low+high) / 2;
  if (A[m] <= x)
    return occurs(A, m, high, x);
  else
    return occurs(A, low, m, x);
}

// Pre: v is sorted in strictly increasing order.
// Returns the number of positions in w that contain elements that also occu
int how_many_in_common(const vector<int>& v,const vector<int>& w) {
  int common = 0;
```

```
  for (int i = 0; i < int(w.size()); ++i)
    if (occurs(v, -1, v.size(), w[i])) ++common;
  return common;
}
```

How many constant columns? X62240_en

Given a matrix M of natural numbers, and a column of M, we say that this column is constant if all its values coincide. For example, consider the following 3×6 matrix:

```
3 1 2 5 4 2
3 5 2 5 4 1
3 1 3 5 4 0
```

Note that column 0 is constant, since all its positions contain a 3. Similarly, column 3 is constant with value 5, and column 4 is constant with value 4.
Write a function that counts the number of constant columns of a given matrix. Use the following C++ code, modify only the parts explicitly indicated.

```
#include <iostream>
#include <vector>

using namespace std;

typedef vector<int> Row;
typedef vector<Row> Matrix;

// reads an n x m integer matrix from the input, n >= 1, m >= 1
Matrix read_matrix(int n, int m) {
    Matrix M(n, Row(m));
    for (int i=0; i < n; ++i)
        for (int j=0; j < m; ++j)
        cin >> M[i][j];
    return M;
}

// Add other functions here if you need them.

int how_many_constant_columns(const Matrix& M)
{
    // implement the function here
}

int main() {
    int n, m;
    while (cin >> n >> m) {
        Matrix M = read_matrix(n, m);
        cout << how_many_constant_columns(M) << endl;
    }
}
```

Exam score: 2.5 **Automatic part:** 100%

Input

The input has several cases. Each case starts with two integers $n, m > 0$ in a first line, followed by n lines with m natural numbers each, that represents a given matrix, and followed by a blank line.

Output

For each case of matrix M of the input, write the number of constant columns of M, followed by a line break.

Sample input

```
2 5
9 5 0 2 1
9 9 0 2 1

3 4
9 2 1 9
3 7 1 9
2 2 1 9

4 4
9 4 7 7
4 5 0 7
6 1 0 7
3 2 0 7

3 6
4 9 3 9 0 8
4 5 4 9 0 8
4 0 7 9 0 8

2 5
6 7 2 1 6
0 7 2 1 6

2 4
9 4 9 0
9 1 7 7

2 4
8 7 9 6
8 6 9 9

6 1
8
8
5
0
9
6

3 2
8 1
8 5
8 8

1 6
9 2 2 9 3 7
```

Sample output

```
4
2
1
4
4
1
2
0
1
6
```

Problem information

Author : PRO1
Generation : 2016-05-28 13:22:17

Point inside a rectangle X82451_en

Given list of pairs ⟨point,rectangle⟩, for each pair we want to know if the point is inside, at the borders, or outside the rectangle. **Complete (and respect) the following code to achieve this goal. Not respecting the code will invalidate your submission, even if it is accepted**

```
#include <iostream>
#include <string>

using namespace std;

// Represents a point by its coordinates x,y.
struct Point {
  int x,y;
};

// Reads a point from the standard input and returns it.
Point read_point()
{
  Point p;
  cin>>p.x>>p.y;
  return p;
}

// Represents a rectangle by the positions its horizontal limits xmin<xmax
// and the positions of its vertical limits ymin<ymax.
struct Rectangle {
  int xmin,ymin,xmax,ymax;
};

// Reads a rectangle from the input and returns it. Assumes that the input form
Rectangle read_rectangle()
{
  Rectangle r;
  cin>>r.xmin>>r.ymin>>r.xmax>>r.ymax;
  return r;
}

// Returns "inside", "border" or "outside" depending on whether
// p is inside, at the border, or outside of r.
string containtment(Point p,Rectangle r)
{
  ...
}

int main()
{
```

```
  ...
}
```

Exam score: 2.5 **Automatic part:** 100%

Input

The first line of the input has an integer $n \geq 1$. Each one of the next n lines has six integers $x, y, x_{min}, y_{min}, x_{max}, y_{max}$ holding $x_{min} < x_{max}$ and $y_{min} < y_{max}$.

Output

For each input line with $x, y, x_{min}, y_{min}, x_{max}, y_{max}$, write "inside", "border" or "outside" depending on whether the point represented by x, y is inside, at the border, or outside the rectangle represented by $x_{min}, y_{min}, x_{max}, y_{max}$, followed by an end of line.

Sample input

```
10
1 -2 -2 0 2 1
-3 1 -3 -4 4 2
-3 3 -5 1 -2 4
1 -3 -4 3 2 4
-5 -3 -3 -2 2 0
-3 4 -3 2 4 3
1 -2 -4 -3 4 -2
4 -4 -1 -1 3 2
-5 0 -4 -5 -2 1
-2 1 -3 -5 1 4
```

Sample output

```
outside
border
inside
outside
outside
outside
border
outside
outside
inside
```

Problem information

Author : Professorat de PRO1
Generation : 2016-05-28 13:30:31

Sort rectangles X61415_en

Write a program that reads a list of rectangles from the input (defined by $x_{min}, y_{min}, x_{max}, y_{max}$), and prints again the same list in the output, but ordering the rectangles by increasing area. In the case of identical area, rectangles with smaller x_{min} go first. In the case of identical area and identical x_{min}, rectangles with smaller y_{min} go first. Finally, if there is still a tie, rectangles with smaller x_{max} go first.
Use the following C++ code, modify only the parts explicitly indicated. It is recomendable to use the library function `sort` to sort the given vector of `Rectangles` in order to avoid the **Time Limit Exceeded**, but you are allowed to implement your own sorting method, with the risk of losing the half of the automatic score, and having a lower manual score, specially if it is slow.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

struct Rectangle {
    int xmin, ymin, xmax, ymax;
};

// read n rectangles from the standard input
vector<Rectangle> read_rectangles(int n) {
    vector<Rectangle> v(n);
    // implement here the rest of the function
    return v;
}

// print the rectangles, end each rectangle followed by a line break.
void print_rectangles(const vector<Rectangle>& v) {
    // implement here the function
}

// Pre: R.xmin<R.xmax and R.ymin<R.ymax.
// returns the area of R.
int area(const Rectangle& R)
{
    // Implement here the function.
}

// returns true if and only if R1 is smaller than R2 (see the statement for a f
bool smaller(const Rectangle& R1, const Rectangle& R2) {
    // implement here the function
}
```

```
// Add other functions if you need them.

int main() {
    int n;
    cin >> n;
    vector<Rectangle> v = read_rectangles(n);
    // Specify correctly the sort call or implement your own sorting method.
    // sort(...);
    print_rectangles(v);
}
```

Exam score: 2.5 **Automatic part:** 50%

Input

The first line has an integer n. The following n lines have four integers each, $x_{min}, y_{min}, x_{max}, y_{max}$ defining a rectangle, where $x_{min} < x_{max}$ and $y_{min} < y_{max}$ hold.

Output

Print the list of the rectangles of the input in the output, with the same format as in the input, but ordered by the following criterias given from higher to lower priority: area, x_{min}, y_{min} and x_{max}.

Sample input

```
11
-1 -1 1 2
0 0 1 1
-1 0 2 2
5 4 6 5
-1 0 2 2
5 1 7 3
-3 5 0 7
0 -5 1 -4
4 -3 7 -2
5 5 7 7
4 -3 5 0
```

Sample output

```
0 -5 1 -4
0 0 1 1
5 4 6 5
4 -3 5 0
4 -3 7 -2
5 1 7 3
5 5 7 7
-3 5 0 7
-1 -1 1 2
-1 0 2 2
-1 0 2 2
```

Problem information

Author : PRO1
Generation : 2016-05-22 16:19:00

Common elements X41129_en

We are given an initial vector v of integers whose elements are sorted in strictly increasing order, that is, $v[i] < v[i+1]$ for each valid positions $i, i+1$ of v. We are also given several cases of vectors w of integers, not necessarily sorted, and whose elements are not necessarily distinct. The goal is to count how many positions in w contain elements that also appear in v.
For example, suppose that $v = [2, 3, 5, 7, 11, 13, 15]$. Then, for the case where $w = [4, 5, 1, 2, 5, 4, 3]$, the answer is 4, as the four positions $w[1] = 5,\ w[3] = 2,\ w[4] = 5,\ w[6] = 3$ are the ones that contain elements that appear also in v.
Your solution should take advantage of the fact that v is sorted in strictly increasing order. Otherwise, you may obtain **Time Limit Exceeded** for the private cases, and thus the half of the automatic score, and a lower manual score. Use the following C++ code, modify only the parts explicitly indicated.

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> read_vector(int n) {
     vector<int> v(n);
     for (int i = 0; i < n; ++i)
         cin >> v[i];
     return v;
}

// add here functions or procedures if you need them

// Pre: v is sorted in strictly increasing order.
// Returns the number of positions in w that contain elements that also occur i
int how_many_in_common(const vector<int>& v,const vector<int>& w) {
     // implement the function here
}

int main() {
    int sz;
    cin>>sz;
    // sz = the size of v
    vector<int> v = read_vector(sz);
    while (cin >> sz) {
          // sz = the size of w
          vector<int> w = read_vector(sz);
          cout << how_many_in_common(v, w) << endl;
    }
}
```

Exam score: 2.5 **Automatic part:** 50%

Input

The input starts with the definition of the vector v. In a first line there is a natural number $n > 0$. In a second line there are the n elements of v, that is $v_0, v_1, \ldots, v_{n-1}$. Next, several alternative cases of definitions of the vector w follow, with the same format as v.

Output

For each case of w, print the number of positions of w that contain elements that also occur in v.

Sample input

```
10
-9 -8 -6 -3 -2 -1 0 5 9 10
5
5 6 6 7 4
8
-1 -8 -5 -5 3 -9 9 -5
3
-7 2 7
8
-9 -3 6 6 5 8 2 4
1
0
7
-8 -8 5 7 9 -3 -2
5
-1 1 1 -9 0
2
-4 1
8
7 -5 9 8 0 4 2 -7
7
-7 7 -1 1 4 -1 4
```

Sample output

```
1
4
0
3
1
6
3
0
2
2
```

Problem information

Author : PRO1
Generation : 2016-05-22 16:22:17