**Code Jam 2008 - Round 1B**

# Analysis: Crop Triangles

The small case was easy to solve using brute force. For each combination of three points you needed to test if the center had integer coordinates. One tricky part was generating the sequence of points, since using 32-bit integers would have resulted in overflow when doing multiplications. One way to get around that problem was to use 64-bit integers.

The observation that helps in solving the large case is that we don't care about the range of the coordinates for the points in the input. We only care about the coordinates modulo 3. We have 9 different classes of points. In bucket[i] we will count the number of points for which x % 3 = i / 3 and y % 3 = i % 3. There are three different ways of choosing 3 points out of the 9 classes of points: three points from the same class, two points from the same class and one from a different class and three points from three different classes.

If the three points are in the same class then it's obvious that the center will have integer coordinates. It is easy to see that there is no triangle with two points in the same class and one point in a different class.

Here is some code that implements this idea:

```
for (int i = 0; i < n; i++) {
  bucket[((int)X0 % 3) * 3 + (int)Y0 % 3]++;
  X0 = (A * X0 + B) % M;
  Y0 = (C * Y0 + D) % M;
}

// The first case.
for (int i = 0; i < 9; i++)
  // We use the formula for n choose 3 so that,
  // we don't use the same point twice or count
  // the same triangle more than once.
  ret += bucket[i] * (bucket[i]-1) * (bucket[i]-2) / 6;

// The third case.
for (int i = 0; i < 9; i++)
  for (int j = i + 1; j < 9; j++)
    for (int k = j + 1; k < 9; k++)
      if (((i / 3) + (j / 3) + (k / 3)) % 3 == 0) &&
          ((i % 3) + (j % 3) + (k % 3)) % 3 == 0)
        ret += bucket[i] * bucket[j] * bucket[k];
cout << "Case #" << prob++ << ": " << ret << endl;
```