Pastebin Link: http://paste.ubuntu.com/24525474/

typedef long long ll;

struct point{

ll x, y;

}C[SIZE];

ll tri\_area(point a, point b, point c){

return (a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y));

}

int n;

/\*

Checks if a point lies within a line segment

No need to write inLineSegment function if lying on

the boundary of the polygon is not considered inside

the polygon

\*/

bool inLineSegment(point a, point b, point p){

ll minx, miny, maxx, maxy;

minx=min(a.x, b.x);

maxx=max(a.x, b.x);

maxy=max(a.y, b.y);

miny=min(a.y, b.y);

if(tri\_area(a, b, p)==0LL &&

p.x>=minx && p.x<=maxx &&

p.y>=miny && p.y<=maxy){

return true;

}

return false;

}

bool inConvexPoly(point p){ //Returns if a point is within a CONVEX polygon

int st=1, ed=n-1, mid;

while((ed-st)>1){ //Complexity: O(lg n)

mid=(st+ed)>>1;

if(tri\_area(C[0], C[mid], p)<0LL) ed=mid;

else st=mid;

}

if(inLineSegment(C[st], C[ed], p)) return true; /\*

^Add this only if being in the

boundary of a polygon is

considered inside

\*/

if(tri\_area(C[0], C[st], p)<0LL) return false;

if(tri\_area(C[st], C[ed], p)<0LL) return false;

if(tri\_area(C[ed], C[0], p)<0LL) return false;

return true;

}