Land division

Submission deadline:	**2011-11-13 23:59:59**	360829.658 sec
Evaluation:	**0.0000**	
Max. assessment:	**5.0000** (Without bonus points)	
Submissions:	0 / 10 Free retries + 20 Penalized retries (-2 % penalty each retry)	
Advices:	0 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)	

Your task is to develop a function which helps the kings with their heritage - the function will help computation how to divide the land into the rightful inheritors.

It is not easy to be a king. Among other duties, the kings must take care of their last will to responsibly assign their property to their inheritors. It has been demonstrated several times that it is not a good idea to leave the entire property to the oldest son - such a decision usually leads to civil wars. Therefore, the king of Convexania decided to divide the land evenly to his two sons. The land of Covexania is of a polygonal shape, moreover, the polygon is convex (as the name of the country indicates). The boundary stones are already in place and it is laborious to move them. Therefore, the king decided that the new border line will simply connect two already existing boundary stones. Thus, the problem is to choose two boundary stones such that the area of the two new lands will be as close to each other as possible.

This assignment requires to implement a function, the function must match the following interface:

```
int divideLand ( double x[], double y[], int n, double * diff );
```

x and y
: are two arrays with X and Y coordinates of the boundary stones. The values `x[i]` and `y[i]` are the x and y coordinates of the `i-th` boundary stone. The order of values in the arrays follows the order of the boundary stones on the borderline. The testing environment guarantees that the values form a convex polygon.

n
: is the number of border stones

diff
: is an output parameter which shall be filled with the difference of the two newly created lands (the smallest difference among all divisions possible),

return value
: the function shall return 1 if it succeeds or 0 if it fails (the land cannot be further divided, i.e. there is less than 4 border stones in the input).

Submit a source file containing the implementation of the `divideLand` function. The source file must contain the function itself and all your supplementary functions needed (called from) the function. On the other hand, the source file shall not contain M `#include` preprocessor directives and `main` function (if the `#include` definitions and `main` function are inside a conditional compile block, they may stay in the submitted file). Use the code below as a basis for your development. If the preprocessor definitions and conditional compilation remain unmodified, the file may be submitted to the Progtest.

```
#ifndef __PROGTEST__
#include <stdio.h>
#include <stdlib.h>
#endif /* __PROGTEST__ */

/* your supplementary functions */

int divideLand ( double x[], double y[], int nr, double * diff )
 {
   /* implementation */
 }

#ifndef __PROGTEST__
int main ( int argc, char * argv [] )
```

```
 {
   /* tests */
 }
#endif /* __PROGTEST__ */
```

Your implementation will be included in a testing environment. There will be both time and memory limits when testing your implementation. The problem does not require much memory, however, it may be time expensive. A solution based on the naive algorithm (test all possible divisions) will pass all tests except the bonus test. To pass the bonus test, the solution must be based on an improved algorithm.

Sample run:

```
int    x;
double diff;
double x0[4] = { 0, 50, 60, 0 };
double y0[4] = { 0, 0, 60, 50 };

double x1[5] = { 0, 5, 15, 20, 10 };
double y1[5] = { 10, 0, 0, 10, 15 };

double x2[3] = { 5, 15, 10 };
double y2[3] = { 0, 0, 15 };

x = divideLand ( x0, y0, 4, &diff ); /* x = 1, diff = 0.000000 */

x = divideLand ( x1, y1, 5, &diff ); /* x = 1, diff = 75.000000 */

x = divideLand ( x2, y2, 3, &diff ); /* x = 0, diff  N/A */
```

Help

- Divide the polygon into triangles to solve the area of the polygon. A good idea is to use a "fan" of triangles. This does not have any further limitations since the polygon is convex here.
- A cross product or Heron's formula may be used to compute the area of a triangle.
- The `diff` value returned may slightly differ from the reference, the threshold is 1e-10 of the polygon area.

Submit: [redacted] **Submit**

Reference