4 Robustness issues: numerical issues, degenerate cases.

4.1 `double` arithmetic

(Assume rounding mode is to the nearest represantable `double`).

4.1.1 Multiplication

For real numbers we have

$$\forall a, b, c \in \mathbb{R}, a, b, c > 0 \qquad a < b \Rightarrow a \cdot c < b \cdot c$$

Now if `a`, `b`, and `c` are three non negative `double` such that `(a<b)` evaluates to `true`.
— Is `a*c<b*c` always `true` ? (Prove or give a counter-example [write numbers in binary])
— Is `a*c<=b*c` always `true` ? (Prove or give a counter-example [write numbers in binary])

4.1.2 Integers in `double`

Let $x_1, x_2, x_3, y_1, y_2, y_3$ integers between -2^b and 2^b.
Find the largest value of b so that you can prove that the expressions
$(x_2 - x_1) * (y3 - y_1) - (x_3 - x_1) * (y_2 - y_1)$
and
$x_2 * y_3 + x_3 * y_1 + x_1 * y_2 - x_3 * y_2 - x_1 * y_3 - x_2 * y_1$
certainly evaluates the same.

4.1.3 A function

What does the following function return when called on a `double` in the open interval $]-2^{51}, 2^{51}[$?

```
double WhoAmI{double x}
  {
    double a = 6755399441055744.0;       // 2^51 + 2^52
    double s = x+0.5+a;
    double r = s-a;
    return r;
  }
```

4.1 Correction:

4.1.1 Multiplication

`a*c<b*c` can be `false`.

$$\begin{array}{rcl} 1.100\ldots0001 \times 1.100\ldots0001 & = & 10.010\ldots001100\ldots0001 \\ & \textit{round to} & 10.010\ldots010 \\ 1.100\ldots0001 \times 1.100\ldots0010 & = & 10.010\ldots01001\ldots0010 \\ & \textit{round to} & 10.010\ldots010 \end{array}$$

`a*c<=b*c` is always `true`.
The true values ac and bc are in the correct order. The nearest reprentable values cannot be swapped.

4.1.2 Integers in `double`

They are both evaluations of the determinant $\begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}$ substracting the first column to the others or using the Sarrus rule. The sign give the orientation predicate.

— The first expression:

$x_2 - x_1$ type expressions use at most $b + 1$ bits

$(x_2 - x_1) * (y3 - y_1)$ type expressions use at most $2b + 2$ bits

$(x_2 - x_1) * (y3 - y_1) - (x_3 - x_1) * (y_2 - y_1)$ uses at most $2b + 3$ bits

— The second expresssion :

$x_2 * y_3$ type expressions use at most $2b$ bits

$x_2 * y_3 + x_3 * y_1 + x_1 * y_2 - x_3 * y_2 - x_1 * y_3 - x_2 * y_1$ use at most $2b + 3$ bits

Thus if $2b + 3 \leq 53$, that is $b \leq 25$, both computations are exact, since double have 53 significant bits. If $b > 25$ rounding errors may creates differences between the evaluations of the two expresssions.

4.1.3 A function

Answer: Rounding to closest integer.

Proof: $2^{52} = 2^{52} + 2^{51} - 2^{51} < x + 0.5 + a < 2^{52} + 2^{51} + 2^{51} = 2^{53}$. So, the value of first significant bit of `s` is $= 2^{52}$, and the value of the 53^{rd} significant bit of `s` is $2^0 = 1$. Since the rounding mode is to closest, `s` becomes the integer closest to `x+0.5+a`. Finally, `r` is the integer that is closest to `x+0.5`, that is integral part of `x+1`.

4.2 Segment intersection

Let S_1 and S_2 be two line segments with endpoints (x_1, y_1), (x'_1, y'_1), (x_2, y_2), and (x'_2, y'_2).

4.2.1 Orientation

Recall the expression of the orientation predicate: `is_ccw`$(x_p, y_p, x_q, y_q, x_r, y_r)$.

4.2.2 Predicate for segment intersections

Write the predicate testing if S_1 and S_2 intersect using calls to `is_ccw`.

4.2 Correction:

4.2.1 Orientation

`is_ccw`$(x_p, y_p, x_q, y_q, x_r, y_r)$

$\delta = (x_q - x_p) * (yr - y_p) - (x_r - x_p) * (y_q - y_p)$;

`return` $(\delta > 0)$;

4.2.2 Predicate for segment intersections

`does_intersect`$(x_1, y_1, x'_1, y'_1, x_2, y_2, x'_2, y'_2)$

`return (` `(` `is_ccw`$(x_1, y_1, x'_1, y'_1, x_2, y_2)$

$\neq$ `is_ccw`$(x_1, y_1, x'_1, y'_1, , x'_2, y'_2)$)

`and (` `is_ccw`$(x_1, y_1, x_2, y_2, x'_2, y'_2)$

$\neq$ `is_ccw`$(x'_1, y'_1, x_2, y_2, x'_2, y'_2)$));

5 Homework 5

5.1