

Pipes
By Duncan Strand
dstrand@zebs.org.uk

Table of Contents

TABLE OF CONTENTS	2
INTRODUCTION	3
THE SQL.....	4
TESTING.....	7
TESTING.....	7
Test1.....	7
Test 2.....	7
Test 3.....	8
Test 4.....	8
Test 5.....	8
Test 6.....	9
Test 7.....	10
THE TEST RESULTS.....	10
CONCLUSION	11
APPENDIX.....	12
SQL SCRIPTS.....	12
Test 1.....	13
Test 2.....	14
Test 3.....	16
Test 4.....	18
Test 5.....	20
Test 6.....	22
Test 7.....	24
Why1	25
Why 2	27
Why 3	29
Final Test	31
APPENDIX B.....	33
TEST GRAPHS	33

Introduction

I will produce a SQL script capable of displaying a table showing the co-ordinates of all the crossovers in a set of data. I will test this script with several sets of data and will, assuming it doesn't work attempt to find out why.

I have (or will) put a copy of this work, including the SQL scripts and test data, on my website at: www.zeps.org.uk/ass/year4sem7/isys3072

The SQL

During the development of the script I kept to one set of test data, this data is test 1 and can with all the other tests be seen in appendix A.

The first task was to create an sql function that can be used to calculate the x coordinate. The function (called xcoord) takes 4 integer inputs for each line. x1 (\$1), x2 (\$2), x3 (\$3), x4 (\$4) for the second line y1 (\$5), y2 (\$6), y3 (\$7), and y4 (\$8)

```
create function xcoord( integer, integer, integer, integer,
integer, integer, integer,integer)
returns double precision
as
begin
return (
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;
```

The function to calculate the y coordinate is:

```
create function ycoord(integer, integer, integer, integer,
integer, integer, integer,integer)
returns double precision
as
begin
return (
( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;
```

The table used to store the co-ordinates of each point in a line is called line_coords.

It is created using:

```
create table line_coords
(
LineNo          integer not null,
LinePart       integer not null,
x1              integer not null,
x2              integer not null,
y1              integer not null,
y2              integer not null,
primary key( LineNo, LinePart )
);
```

The first set of test data is:

LineNo	LinePart	x1	x2	y1	y2
1	1	10	40	10	40
1	2	40	50	40	10
2	1	10	30	20	20
2	2	30	30	20	10
3	1	40	50	30	30

With this test data there are 2 crossovers, at co-ordinates 20,20 (lines 1 and 2) and at (lines 1 and 3)

To insert the above data into the table:

```
insert into line_coords values ( 1, 1, 10, 40, 10, 40 );
insert into line_coords values ( 1, 2, 40, 50, 40, 10 );
insert into line_coords values ( 2, 1, 10, 30, 20, 20 );
insert into line_coords values ( 2, 2, 30, 30, 20, 10 );
insert into line_coords values ( 3, 1, 40, 50, 30, 30 );
```

We now need to convert these values into something that can be used by the 2D Blade. Although the Path datatype is the most accurate representation of the data it does not allow me to easily create it from the available data and tables, nor does it allow me to determine which line segments are overlapping. Because of this I have opted for the Line (Lseg) datatype.

```
create table pipes
(
  LineNo          integer not null,
  LinePart        integer not null,
  LineCoords      Lseg not null,
  primary key( LineNo, LinePart )
);
```

We need something that takes the co-ordinates from the line_coords table, converting each set of co-ordinates into Lseg. This query performs that function

```
insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2,
y2 ) from line_coords;
```

Now using the pipes table we want to extract all the lines that have a crossover into a separate table called crossovers. For the moment I'm not concerned with the coordinate of the crossover, just the existence of one. To determine if two lines cross we can use the Overlaps function.

The overlaps table is defined as:

```
create table PipeOverlap
(
  LineNo1         integer not null,
  LinePart1        integer not null,
  LineCoords1      Lseg not null,
  LineNo2          integer not null,
  LinePart2        integer not null,
  LineCoords2      Lseg not null,
  primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);
```

To list all the intersecting lines the following query is used

```

select * from pipes p1,pipes p2 where
Intersects(p1.LineCoords, p2.LineCoords ) and p1.LineNo <>
p2.LineNo;

```

This lists all the intersecting lines where the line numbers is different. This is allowed, as we know that no line may cross itself.

To insert this into the PipeOverlap table:

```

insert into PipeOverlap
select * from pipes p1,pipes p2 where
Intersects(p1.LineCoords, p2.LineCoords ) and p1.LineNo <>
p2.LineNo;

```

We now have a table containing a list of all the intersecting pipes, all we need now is a table containing the co-ordinates of the actually crossover, the line no and line part.

Before we can do that however we need a way of getting the co-ordinates of the lines, these values are still stored in the line_coords table. We also need the co-ordinates of both intersecting lines in the same row, so that they may later be passed to the calculating functions.

We now have a table containing all the pipes that cross, but we cannot get the co-ordinates of the pipes from this table, we need a query that creates a table using the data in the above table but also getting the lines coordinates from the line_coords table.

The query to do this is:

```

create table crossCoords as select lc.LineNo, lc.LinePart,
lc.x1, lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP,
lc2.x1 x3, lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart
AND po.LineNo2 = lc2.LineNo AND po.LinePart2 =
lc2.LinePart;

```

Careful examination of this table shows that the data is duplicated, I do not consider this to be a serious problem.

What is needed now is a query to call the x and y co-ordinates functions and insert those values into a table called crossovers. The query is:

```

create table crossovers as select LineNo, LinePart, LN
LineNo2, LP LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4),
ycoord(x1,x2,x3, x4, y1, y2, y3, y4)
from crossCoords c;

```

To display the results:

```

select * from crossovers;

```

To clean up all the tables:

```

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer,
integer, integer, integer,integer);
drop function ycoord(integer, integer, integer, integer,
integer, integer, integer,integer);

```

Testing

After finally getting the co-ordinates of the crossovers in the first test data I proceeded to test the code with more data, each test designed to test certain circumstances.

- Test2 Points of a line intersected by another line.
- Test3 Lines repeatedly crossing, 2 pipes vertical aligned, and fractional co-ordinates.
- Test4 Identical to test 3, but with out the vertical line.
- Test5 Lots of intersections.
- Test6 More lines and therefore lots of intersections.
- Test7 A line crossing itself.

Testing

I will compare the results generated by my SQL script and by calculating the co-ordinates manually (well, entering the co-ordinates into a little program I wrote to make it much easier, its available on the website)

Test1

The expected results are:

Line	Part	Line2	Part2	X	Y
1	1	2	2	20	20
1	2	3	1	43.333	30

The results from Illustra are:

LineNo	LinePart	LineNo2	LinePart2	xcoord	ycoord
2	1	1	1	20	20
3	1	1	2	43	30
1	1	2	1	20	20
1	2	3	1	43	30

These results are as expected as floating point values are not supported within the SQL script.

Test 2

The expected results are:

Line	Part	Line2	Part2	X	Y
1	1	3	6	16.666	16.666
1	2	3	2	30	30
1	3	3	1	41.428	35.714
2	4	1	1	20	20
2	4	3	6	15	20

The results from Illustra are:

LineNo	LinePart	LineNo2	LinePart2	xcoord	ycoord
2	4	1	1	20	20
3	6	1	1	16	16
2	4	1	2	20	20
3	1	1	2	40	40
3	2	1	2	30	30
3	1	1	3	40	40
1	1	2	4	20	20
1	2	2	4	20	20
3	6	2	4	15	20
1	2	3	1	40	40
1	3	3	1	40	40
1	2	3	2	30	30
1	1	3	6	16	16
2	4	3	6	15	20

The reason for Illustra giving so many results is because of the intersections at 20, 20, and 30, 30. At these locations part of a line ends and the next part starts causing both parts of the line to have an intersection. Technically Illustra is correct, but its is producing duplicate data and what's happening isn't immediately obvious.

Test 3

There are intersections at: (ignoring the intersection of line 1 part 1 and line 2 part 2)

Line	Part	Line2	Part2	X	Y
1	2	3	3	~15.163	~32.459
1	3	3	3	~35.294	~33.218
1	3	3	2	~40.065	~25.686
1	3	3	1	~47.166	~47.361
1	4	3	1	Divide by 0	Divide by 0
1	4	3	2	Divide by 0	Divide by 0
1	4	3	3	Divide by 0	Divide by 0
2	2	3	3	Divide by 0	Divide by 0

(all values are approximate as the program I'm using to calculate these values does not allow floating point numbers to be entered)

As expected this test failed for two reasons. The vertical intersection and the use of floating point numbers (ie 30.5). To solve the problem with the numbers I changed the co-ordinate integer values in the line_coords table to floats, this also required that I changed the xcoord and ycoord functions to take floats instead of integers. This gave the error "X22003:data exception: numeric value inf out of range for real", I also tried changing the parameters to a double precision, but this still resulted in the same error.

Test 4

There are intersections at:

Line	Part	Line2	Part2	X	Y
1	1	2	2	Divide by 0	Divide by 0
1	2	3	3	~15.163	~32.459
1	3	3	3	~35.294	~33.218
1	3	3	2	~40.065	~25.686
1	3	3	1	~47.166	~47.361
1	4	3	1	Divide by 0	Divide by 0
1	4	3	2	Divide by 0	Divide by 0
1	4	3	3	Divide by 0	Divide by 0
2	2	3	3	Divide by 0	Divide by 0

(all values are approximate as the program I'm using to calculate these values does not allow floating point numbers to be entered)

Because this test involves floating point values after the previous test I expected this test to fail. Because one of the purposes of this test was to run the data from test 3 but without the vertical line intersection I modified the test by removing the floating values and rounding them up. This modified test resulted in a divide by 0 error

Test 5

Line	Part	Line2	Part2	X	Y
1	1	2	1	Divide by 0	Divide by 0
1	1	2	3	Divide by 0	Divide by 0
1	1	2	5	Divide by 0	Divide by 0
1	1	3	1	17.727	50
1	1	2	7	Divide by 0	Divide by 0
1	3	2	7	Divide by 0	Divide by 0
1	3	2	5	Divide by 0	Divide by 0
1	3	3	1	10.454	40
1	3	2	3	Divide by 0	Divide by 0

1	3	2	1	Divide by 0	Divide by 0
1	5	2	1	Divide by 0	Divide by 0
1	5	2	3	Divide by 0	Divide by 0
1	5	2	5	Divide by 0	Divide by 0
1	5	2	7	Divide by 0	Divide by 0
2	3	3	1	Divide by 0	Divide by 0
2	5	3	1	Divide by 0	Divide by 0
2	7	3	1	Divide by 0	Divide by 0

Using the SQL script this test resulted in a divide by 0 error.

Test 6

Line	Part	Line2	Part2	X	Y
1	1	2	1	Divide by 0	Divide by 0
1	1	2	3	Divide by 0	Divide by 0
1	1	2	5	Divide by 0	Divide by 0
1	1	3	1	17.727	50
1	1	2	7	Divide by 0	Divide by 0
1	1	4	7	25.2	50
1	1	4	5	32	50
1	1	4	2	34.347	50
1	3	4	2	36.521	40
1	3	4	5	34.5	40
1	3	2	7	Divide by 0	Divide by 0
1	3	2	5	Divide by 0	Divide by 0
1	3	3	1	10.454	40
1	3	2	3	Divide by 0	Divide by 0
1	3	2	1	Divide by 0	Divide by 0
1	5	2	1	Divide by 0	Divide by 0
1	5	2	3	Divide by 0	Divide by 0
1	5	2	5	Divide by 0	Divide by 0
1	5	3	2	19.333	35
1	5	2	7	Divide by 0	Divide by 0
1	5	4	5	35.75	35
1	5	4	2	37.608	35
2	1	3	6	Divide by 0	Divide by 0
2	1	4	4	Divide by 0	Divide by 0
2	1	4	3	Divide by 0	Divide by 0
2	3	4	3	Divide by 0	Divide by 0
2	3	4	4	Divide by 0	Divide by 0
2	3	3	2	Divide by 0	Divide by 0
2	3	3	1	Divide by 0	Divide by 0
2	3	3	6	Divide by 0	Divide by 0
2	5	3	1	Divide by 0	Divide by 0
2	5	3	2	Divide by 0	Divide by 0
2	5	4	4	Divide by 0	Divide by 0
2	5	4	3	Divide by 0	Divide by 0
2	7	4	3	Divide by 0	Divide by 0
2	7	4	4	Divide by 0	Divide by 0
2	7	3	2	Divide by 0	Divide by 0
2	7	3	1	Divide by 0	Divide by 0
3	2	4	5	36.549	31.801
3	2	4	2	38.839	29.336
3	3	4	2	39.291	27.260
3	3	4	3	38.083	24.294
3	5	4	3	Divide by 0	Divide by 0
3	5	4	4	Divide by 0	Divide by 0

Using the SQL script this test resulted in a divide by 0 error.

Test 7

This found no intersecting pipes, and because of the vertical line would result in a divide by 0 error.

The test results

Because of the poor test results I attempted to find out why it failed on almost all of the tests. I started by creating more simple tests and running them.

I created some new test data (called Why1) and ran the test. It worked and gave the following results:

LineNo	LinePart	LineNo2	LinePart2	xcoord	ycoord
2	1	1	1	20	20
3	1	1	1	15	15
1	1	2	1	20	20
3	1	2	1	16	20
1	1	3	1	15	15
2	1	3	1	16	20

After examining the data and the calculations performed on it I discovered that just having a vertical line with an also causes problems (divide by 0). I modified the above test so that values x2 of pipe 3 was 15 and as expected got a divide by 0.

I modified test 4 by removing the vertical lines (see test Why2). It resulted in a divide by 0.

I decided to test if having the same part of a line intersected by more than 1 different line caused any problems this test, called Why3 resulted in:

LineNo	LinePart	LineNo2	LinePart2	xcoord	ycoord
2	1	1	1	7	10
3	1	1	1	7	10
1	1	2	1	7	10
1	1	3	1	7	10

Finally I decided to create a final test where there are no vertical lines and no parts of a line that are crossed twice (see final test), this resulted with a divide by 0.

Line	Part	Line2	Part2	X	Y
1	1	2	2	5.702	7.531
1	2	2	3	3.875	8
1	3	4	4	2.875	7.75
2	1	3	6	18.048	15.121
3	1	5	6	21.777	10.888
3	2	5	3	25.419	14.129
3	3	4	1	22.307	20.538
3	4	4	2	18.888	22.777
3	5	5	1	11.818	17.545

Conclusion

Obviously there are problems with the SQL, it doesn't handle vertical lines, or floating point coordinates. It is also very easy when entering the data to enter it incorrectly, so that two line segments don't actually meet.

It also outputs the same data twice, in test 1 there are only two intersections but four are displayed these are the same data one showing line 1 intersecting line 2, and the other showing line 2 intersecting line 1.

Appendix

SQL Scripts

Test 1

```
create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return      (
      ( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
      *
      ( ( $8 - $7 ) / ( $4 - $3 ) ) )
      -
      ( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
      *
      ( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
      /
      ( ( ( $8 - $7 ) / ( $4 - $3 ) )
      -
      ( ( $6 - $5 ) / ( $2 - $1 ) )
      );
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return      (
      ( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
      ( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
      )
      /
      (
      ( ( $6 - $5 ) / ( $2-$1 ) )
      -
      ( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
      LineNo          integer not null,
      LinePart        integer not null,
      x1              integer not null,
      x2              integer not null,
      y1              integer not null,
      y2              integer not null,
      primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 10, 40, 10, 40 );
insert into line_coords values ( 1, 2, 40, 50, 40, 10 );
insert into line_coords values ( 2, 1, 10, 30, 20, 20 );
insert into line_coords values ( 2, 2, 30, 30, 20, 10 );
insert into line_coords values ( 3, 1, 40, 50, 30, 30 );

create table pipes
(
      LineNo          integer not null,
      LinePart        integer not null,
      LineCoords      Lseg not null,
      primary key( LineNo, LinePart )
```

```

);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1          integer not null,
    LinePart1        integer not null,
    LineCoords1      Lseg not null,
    LineNo2          integer not null,
    LinePart2        integer not null,
    LineCoords2      Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 2

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
    ( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
    *
    ( ( $8 - $7 ) / ( $4 - $3 ) ) )
    -
    ( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
    *
    ( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
    /
    ( ( ( $8 - $7 ) / ( $4 - $3 ) )
    -

```

```

        ( ( $6 - $5 ) / ( $2 - $1 ) )
    );
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return    (
        ( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
        ( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
        )
        /
        (
        ( ( $6 - $5 ) / ( $2-$1 ) )
        -
        ( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
    LineNo            integer not null,
    LinePart         integer not null,
    x1                integer not null,
    y1                integer not null,
    x2                integer not null,
    y2                integer not null,
    primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 10, 10, 20, 20 );
insert into line_coords values ( 1, 2, 20, 20, 40, 40 );
insert into line_coords values ( 1, 3, 40, 40, 50, 10 );

insert into line_coords values ( 2, 1, 40, 10, 40, 20 );
insert into line_coords values ( 2, 2, 40, 20, 30, 10 );
insert into line_coords values ( 2, 3, 30, 10, 30, 20 );
insert into line_coords values ( 2, 4, 30, 20, 10, 20 );

insert into line_coords values ( 3, 1, 50, 40, 30, 30 );
insert into line_coords values ( 3, 2, 30, 30, 20, 30 );
insert into line_coords values ( 3, 3, 20, 30, 30, 40 );
insert into line_coords values ( 3, 4, 30, 40, 20, 40 );
insert into line_coords values ( 3, 5, 20, 40, 10, 30 );
insert into line_coords values ( 3, 6, 10, 30, 20, 10 );

create table pipes
(
    LineNo            integer not null,
    LinePart         integer not null,
    LineCoords       Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1           integer not null,

```

```

        LinePart1    integer not null,
        LineCoords1 Lseg not null,
        LineNo2      integer not null,
        LinePart2    integer not null,
        LineCoords2 Lseg not null,
        primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
    );

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 3

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer, integer )
returns double precision
as
begin
return
(
( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer, integer )
returns double precision
as

```

```

begin
return  (
        ( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
        ( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
        )
        /
        (
        ( ( $6 - $5 ) / ( $2-$1 ) )
        -
        ( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
    LineNo          float not null,
    LinePart        float not null,
    x1              float not null,
    y1              float not null,
    x2              float not null,
    y2              float not null,
    primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 10, 15, 10, 30 );
insert into line_coords values ( 1, 2, 10, 30, 30.5, 40 );
insert into line_coords values ( 1, 3, 30.5, 40, 50, 10 );
insert into line_coords values ( 1, 4, 50, 10, 50, 40 );

insert into line_coords values ( 2, 1, .5, .5, 10, 10 );
insert into line_coords values ( 2, 2, 10, 10, 10, 40 );

insert into line_coords values ( 3, 1, 55, 10, 33, 22 );
insert into line_coords values ( 3, 2, 33, 22, 56, 34 );
insert into line_coords values ( 3, 3, 56, 34, 3, 32 );

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1         integer not null,
    LinePart1       integer not null,
    LineCoords1     Lseg not null,
    LineNo2         integer not null,
    LinePart2       integer not null,
    LineCoords2     Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

```

```

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 4

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
(
-
(
(
(
$6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
(
(
$8 - $7 ) / ( $4 - $3 ) ) )
-
(
-
(
(
$8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
(
(
$6 - $5 ) / ( $2 - $1 ) ) ) )
/
(
(
(
$8 - $7 ) / ( $4 - $3 ) )
-
(
(
$6 - $5 ) / ( $2 - $1 ) )
)
);
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
(
-
(
(
(
$8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
(
-
(
(
$6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
(
(
$6 - $5 ) / ( $2-$1 ) )
-
(
(
$8 - $7 ) / ( $4 - $3 ) ) ) );
end;

```

```

create table line_coords
(
    LineNo          integer not null,
    LinePart        integer not null,
    x1              integer not null,
    y1              integer not null,
    x2              integer not null,
    y2              integer not null,
    primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 11, 15, 10, 30 );
insert into line_coords values ( 1, 2, 10, 30, 30.5, 40 );
insert into line_coords values ( 1, 3, 30.5, 40, 50, 10 );
insert into line_coords values ( 1, 4, 50, 10, 50, 40 );

insert into line_coords values ( 2, 1, .5, .5, 10, 10 );
insert into line_coords values ( 2, 2, 10, 10, 10, 40 );

insert into line_coords values ( 3, 1, 55, 10, 33, 22 );
insert into line_coords values ( 3, 2, 33, 22, 56, 34 );
insert into line_coords values ( 3, 3, 56, 34, 3, 32 );

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1         integer not null,
    LinePart1       integer not null,
    LineCoords1     Lseg not null,
    LineNo2         integer not null,
    LinePart2       integer not null,
    LineCoords2     Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

```

```

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 5

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
LineNo integer not null,
LinePart integer not null,
x1 integer not null,
y1 integer not null,
x2 integer not null,
y2 integer not null,
primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 5, 50, 50, 50 );

```

```

insert into line_coords values ( 1, 2, 50, 50, 50, 40 );
insert into line_coords values ( 1, 3, 50, 40, 5, 40 );
insert into line_coords values ( 1, 4, 5, 40, 5, 35 );
insert into line_coords values ( 1, 5, 5, 35, 55, 35 );

insert into line_coords values ( 2, 1, 6, 51, 6, 12 );
insert into line_coords values ( 2, 2, 6, 12, 10, 11 );
insert into line_coords values ( 2, 3, 10, 11, 10, 52 );
insert into line_coords values ( 2, 4, 10, 52, 15, 53 );
insert into line_coords values ( 2, 5, 15, 53, 15, 10 );
insert into line_coords values ( 2, 6, 15, 10, 20, 9 );
insert into line_coords values ( 2, 7, 20, 9, 20, 54 );

insert into line_coords values ( 3, 1, 25, 60, 9, 28 );
insert into line_coords values ( 2, 1, 25, 60, 9, 28);

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1          integer not null,
    LinePart1        integer not null,
    LineCoords1      Lseg not null,
    LineNo2          integer not null,
    LinePart2        integer not null,
    LineCoords2      Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;

```

```

drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 6

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return      (
      ( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
      *
      ( ( $8 - $7 ) / ( $4 - $3 ) ) )
      -
      ( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
      *
      ( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
      /
      ( ( ( $8 - $7 ) / ( $4 - $3 ) )
      -
      ( ( $6 - $5 ) / ( $2 - $1 ) )
      );
end;

```

```

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return      (
      ( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
      ( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
      )
      /
      (
      ( ( $6 - $5 ) / ( $2-$1 ) )
      -
      ( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

```

```

create table line_coords
(
      LineNo          integer not null,
      LinePart       integer not null,
      x1              integer not null,
      y1              integer not null,
      x2              integer not null,
      y2              integer not null,
      primary key( LineNo, LinePart )
);

```

```

insert into line_coords values ( 1, 1, 5, 50, 50, 50 );
insert into line_coords values ( 1, 2, 50, 50, 50, 40 );
insert into line_coords values ( 1, 3, 50, 40, 5, 40 );
insert into line_coords values ( 1, 4, 5, 40, 5, 35 );
insert into line_coords values ( 1, 5, 5, 35, 55, 35 );

insert into line_coords values ( 2, 1, 6, 51, 6, 12 );
insert into line_coords values ( 2, 2, 6, 12, 10, 11 );

```

```

insert into line_coords values ( 2, 3, 10, 11, 10, 52 );
insert into line_coords values ( 2, 4, 10, 52, 15, 53 );
insert into line_coords values ( 2, 5, 15, 53, 15, 10 );
insert into line_coords values ( 2, 6, 15, 10, 20, 9 );
insert into line_coords values ( 2, 7, 20, 9, 20, 54 );

insert into line_coords values ( 3, 1, 25, 60, 9, 38 );
insert into line_coords values ( 3, 2, 9, 38, 40, 29 );
insert into line_coords values ( 3, 3, 40, 29, 29, 2 );
insert into line_coords values ( 3, 4, 29, 2, 2, 2 );
insert into line_coords values ( 3, 5, 2, 2, 2, 40 );
insert into line_coords values ( 3, 6, 2, 40, 11, 45 );

insert into line_coords values ( 4, 1, 45, 70, 30, 70 );
insert into line_coords values ( 4, 2, 30, 70, 40, 24 );
insert into line_coords values ( 4, 3, 40, 24, 1, 30 );
insert into line_coords values ( 4, 4, 1, 30, 37, 30 );
insert into line_coords values ( 4, 5, 37, 30, 27, 70 );
insert into line_coords values ( 4, 6, 27, 70, 26, 70 );
insert into line_coords values ( 4, 7, 26, 70, 25, 45 );

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1          integer not null,
    LinePart1        integer not null,
    LineCoords1      Lseg not null,
    LineNo2          integer not null,
    LinePart2        integer not null,
    LineCoords2      Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

```

```

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Test 7

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision

```

```

as
begin
return
( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

```

```

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision

```

```

as
begin
return
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

```

```

create table line_coords

```

```

(
LineNo          integer not null,
LinePart        integer not null,
x1              integer not null,
y1              integer not null,
x2              integer not null,
y2              integer not null,
primary key( LineNo, LinePart )
);

```

```

insert into line_coords values ( 1, 1, 5, 5, 5, 8 );
insert into line_coords values ( 1, 2, 5, 8, 2, 8 );
insert into line_coords values ( 1, 3, 2, 8, 2, 6 );

```

```

insert into line_coords values ( 1, 4, 2, 6, 7, 6 );

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);
insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1          integer not null,
    LinePart1        integer not null,
    LineCoords1      Lseg not null,
    LineNo2          integer not null,
    LinePart2        integer not null,
    LineCoords2      Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Why1

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
(
    ( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
    *
    ( ( $8 - $7 ) / ( $4 - $3 ) ) )

```

```

-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return (
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
LineNo          integer not null,
LinePart       integer not null,
x1              integer not null,
y1              integer not null,
x2              integer not null,
y2              integer not null,
primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 10, 10, 20, 20 );
insert into line_coords values ( 2, 1, 10, 20, 30, 10 );
insert into line_coords values ( 3, 1, 15, 10, 16, 20 );

create table pipes
(
LineNo          integer not null,
LinePart       integer not null,
LineCoords     Lseg not null,
primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
LineNo1         integer not null,
LinePart1      integer not null,
LineCoords1    Lseg not null,
LineNo2         integer not null,
LinePart2      integer not null,
LineCoords2    Lseg not null,

```

```

        primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
    );

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Why 2

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
    (
        ( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
        *
        ( ( $8 - $7 ) / ( $4 - $3 ) ) )
        -
        ( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
        *
        ( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
        /
        ( ( ( $8 - $7 ) / ( $4 - $3 ) )
        -
        ( ( $6 - $5 ) / ( $2 - $1 ) )
        );
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as
begin
return
    (
        ( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
        ( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
        )

```

```

/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_coords
(
    LineNo      integer not null,
    LinePart    integer not null,
    x1          integer not null,
    y1          integer not null,
    x2          integer not null,
    y2          integer not null,
    primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 11, 15, 10, 30 );
insert into line_coords values ( 1, 2, 10, 30, 30, 40 );
insert into line_coords values ( 1, 3, 30, 40, 50, 10 );

insert into line_coords values ( 2, 1, 55, 10, 33, 22 );
insert into line_coords values ( 2, 2, 33, 22, 56, 34 );
insert into line_coords values ( 2, 3, 56, 34, 3, 32 );

create table pipes
(
    LineNo      integer not null,
    LinePart    integer not null,
    LineCoords  Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1     integer not null,
    LinePart1   integer not null,
    LineCoords1 Lseg not null,
    LineNo2     integer not null,
    LinePart2   integer not null,
    LineCoords2 Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

```

```

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

```

```

select * from crossovers;

```

```

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Why 3

```

create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as

```

```

begin
return
(
( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

```

```

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision
as

```

```

begin
return
(
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

```

```

create table line_coords
(
LineNo      integer not null,
LinePart    integer not null,
x1          integer not null,
y1          integer not null,
x2          integer not null,

```

```

        y2          integer not null,
        primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 5, 10, 10, 10 );
insert into line_coords values ( 2, 1, 5, 8, 7, 11 );
insert into line_coords values ( 3, 1, 7, 8, 8, 11 );

create table pipes
(
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1          integer not null,
    LinePart1        integer not null,
    LineCoords1      Lseg not null,
    LineNo2          integer not null,
    LinePart2        integer not null,
    LineCoords2      Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

```

Final Test

```
create function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision as
begin
return (
( ( - ( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
*
( ( $8 - $7 ) / ( $4 - $3 ) ) )
-
( ( - ( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 )
*
( ( $6 - $5 ) / ( $2 - $1 ) ) ) )
/
( ( ( $8 - $7 ) / ( $4 - $3 ) )
-
( ( $6 - $5 ) / ( $2 - $1 ) )
);
end;

create function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer)
returns double precision as
begin
return (
( -( ( ( $8 - $7 ) / ( $4 - $3 ) ) * $3 ) + $7 ) -
( -( ( ( $6 - $5 ) / ( $2 - $1 ) ) * $1 ) + $5 )
)
/
(
( ( $6 - $5 ) / ( $2-$1 ) )
-
( ( $8 - $7 ) / ( $4 - $3 ) ) );
end;

create table line_cords (
LineNo          integer not null,
LinePart        integer not null,
x1              integer not null,
y1              integer not null,
x2              integer not null,
y2              integer not null,
primary key( LineNo, LinePart )
);

insert into line_coords values ( 1, 1, 8, 6, 5, 8 );
insert into line_coords values ( 1, 2, 5, 8, 3, 8 );
insert into line_coords values ( 1, 3, 3, 8, 2, 6 );
insert into line_coords values ( 1, 4, 2, 6, 7, 6 );

insert into line_coords values ( 2, 1, 15, 20, 20, 12 );
insert into line_coords values ( 2, 2, 20, 12, 4, 7 );
insert into line_coords values ( 2, 3, 4, 7, 3, 15 );

insert into line_coords values ( 3, 1, 20, 10, 24, 12 );
insert into line_coords values ( 3, 2, 24, 12, 26, 15 );
insert into line_coords values ( 3, 3, 26, 15, 20, 24 );
insert into line_coords values ( 3, 4, 20, 24, 10, 23 );
insert into line_coords values ( 3, 5, 10, 23, 13, 14 );
insert into line_coords values ( 3, 6, 13, 14, 22, 16 );
```

```

insert into line_coords values ( 4, 1, 25, 20, 20, 21 );
insert into line_coords values ( 4, 2, 20, 21, 15, 29 );
insert into line_coords values ( 4, 3, 15, 29, 1, 19 );
insert into line_coords values ( 4, 4, 1, 19, 3, 7 );

insert into line_coords values ( 5, 1, 10, 15, 15, 22 );
insert into line_coords values ( 5, 2, 15, 22, 20, 18 );
insert into line_coords values ( 5, 3, 20, 18, 27, 13 );
insert into line_coords values ( 5, 4, 27, 13, 20, 5 );
insert into line_coords values ( 5, 5, 20, 5, 22, 10 );
insert into line_coords values ( 5, 6, 22, 10, 21, 14 );

create table pipes (
    LineNo          integer not null,
    LinePart        integer not null,
    LineCoords      Lseg not null,
    primary key( LineNo, LinePart )
);

insert into pipes select LineNo, LinePart, Lseg(x1, y1, x2, y2 ) from
line_coords;

create table PipeOverlap
(
    LineNo1         integer not null,
    LinePart1       integer not null,
    LineCoords1     Lseg not null,
    LineNo2         integer not null,
    LinePart2       integer not null,
    LineCoords2     Lseg not null,
    primary key( LineNo1, LinePart1, LineNo2, LinePart2 )
);

insert into PipeOverlap
select * from pipes p1,pipes p2 where Intersects(p1.LineCoords,
p2.LineCoords ) and p1.LineNo <> p2.LineNo;

create table crossCoords as select lc.LineNo, lc.LinePart, lc.x1,
lc.x2, lc.y1, lc.y2, lc2.LineNo LN, lc2.LinePart LP, lc2.x1 x3,
lc2.x2 x4, lc2.y1 y3, lc2.y2 y4
from PipeOverlap po, line_coords lc, line_coords lc2
where po.LineNo1 = lc.LineNo AND po.LinePart1 = lc.LinePart AND
po.LineNo2 = lc2.LineNo AND po.LinePart2 = lc2.LinePart;

create table crossovers as select LineNo, LinePart, LN LineNo2, LP
LinePart2, xcoord(x1,x2,x3,x4,y1,y2,y3,y4), ycoord(x1,x2,x3, x4, y1,
y2, y3, y4)
from crossCoords c;

select * from crossovers;

drop table line_coords;
drop table pipes;
drop table PipeOverlap;
drop table crossCoords;
drop table crossovers;
drop function xcoord( integer, integer, integer, integer, integer,
integer, integer,integer);
drop function ycoord(integer, integer, integer, integer, integer,
integer, integer,integer);

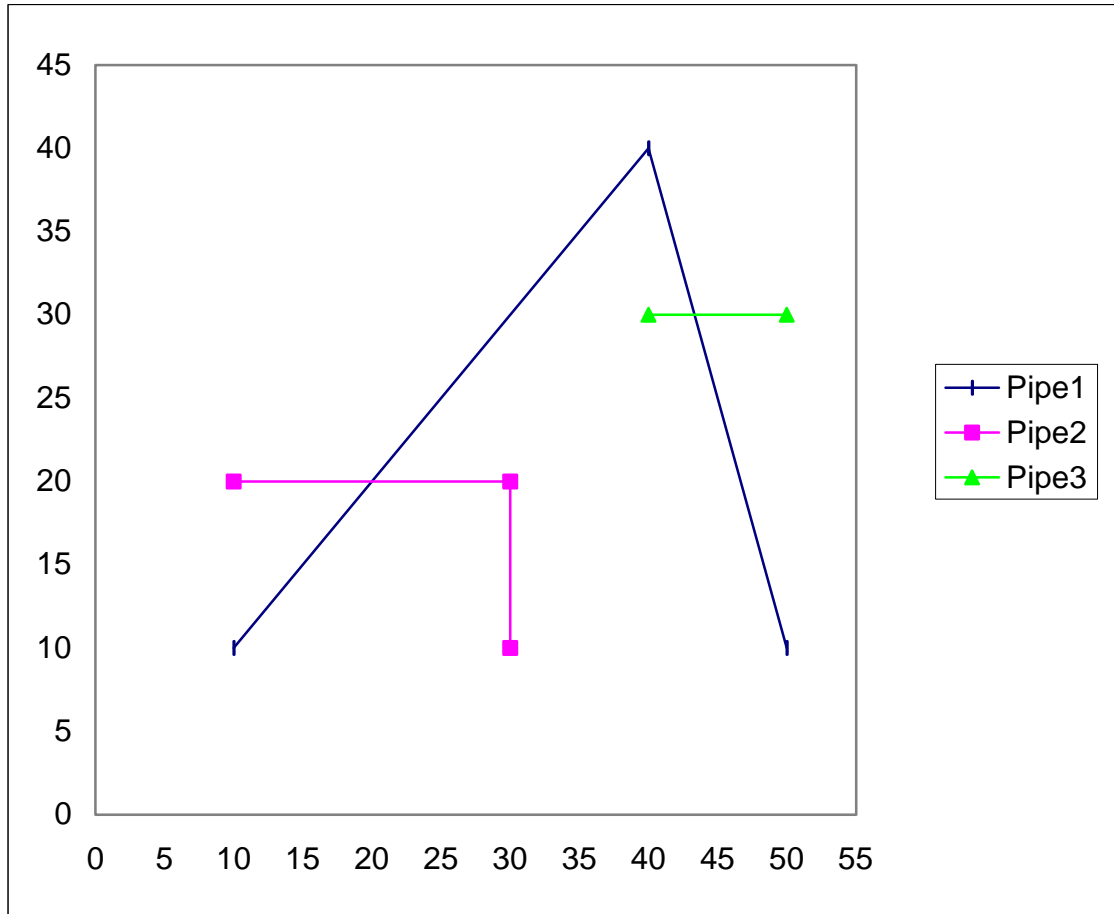
```

Appendix B

Test Graphs

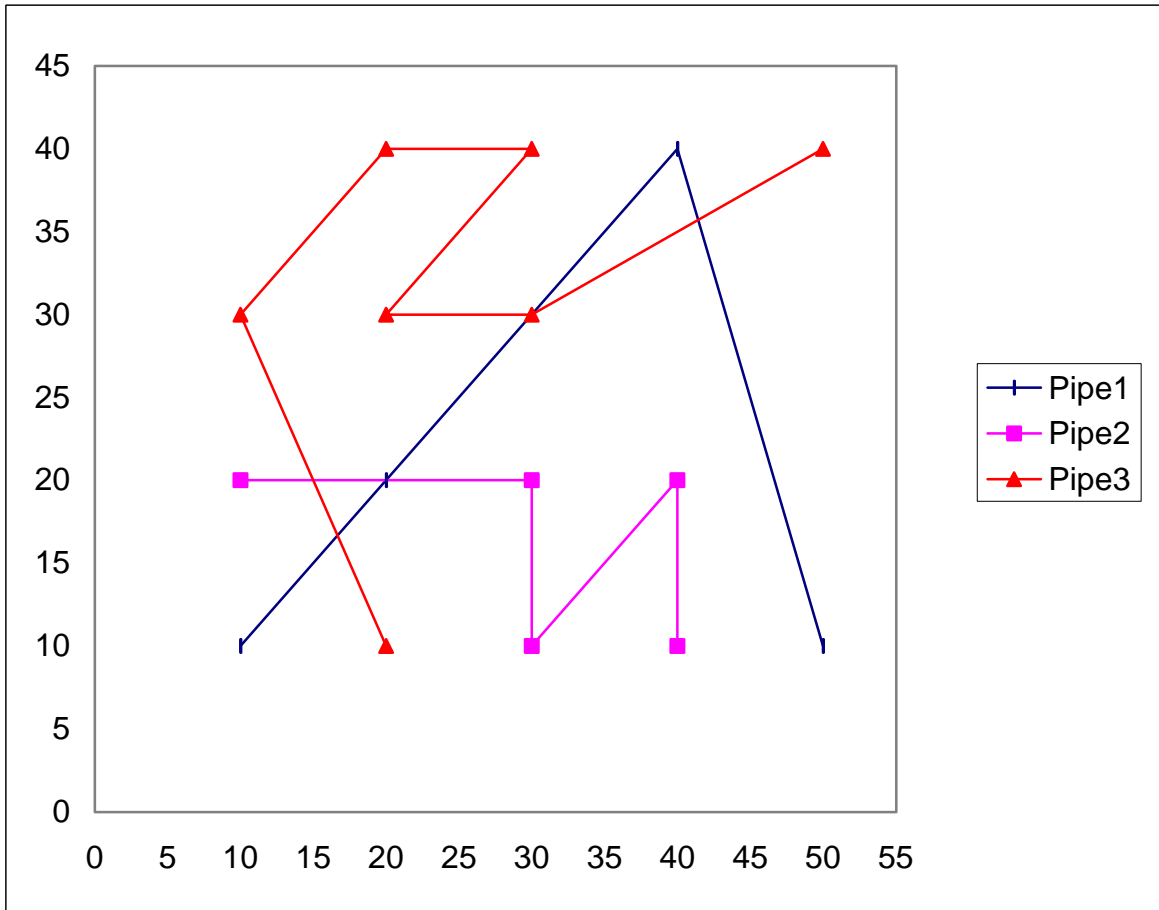
1st Test

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
10	10	30	10	40	30
40	40	30	20	50	30
50	10	10	20		



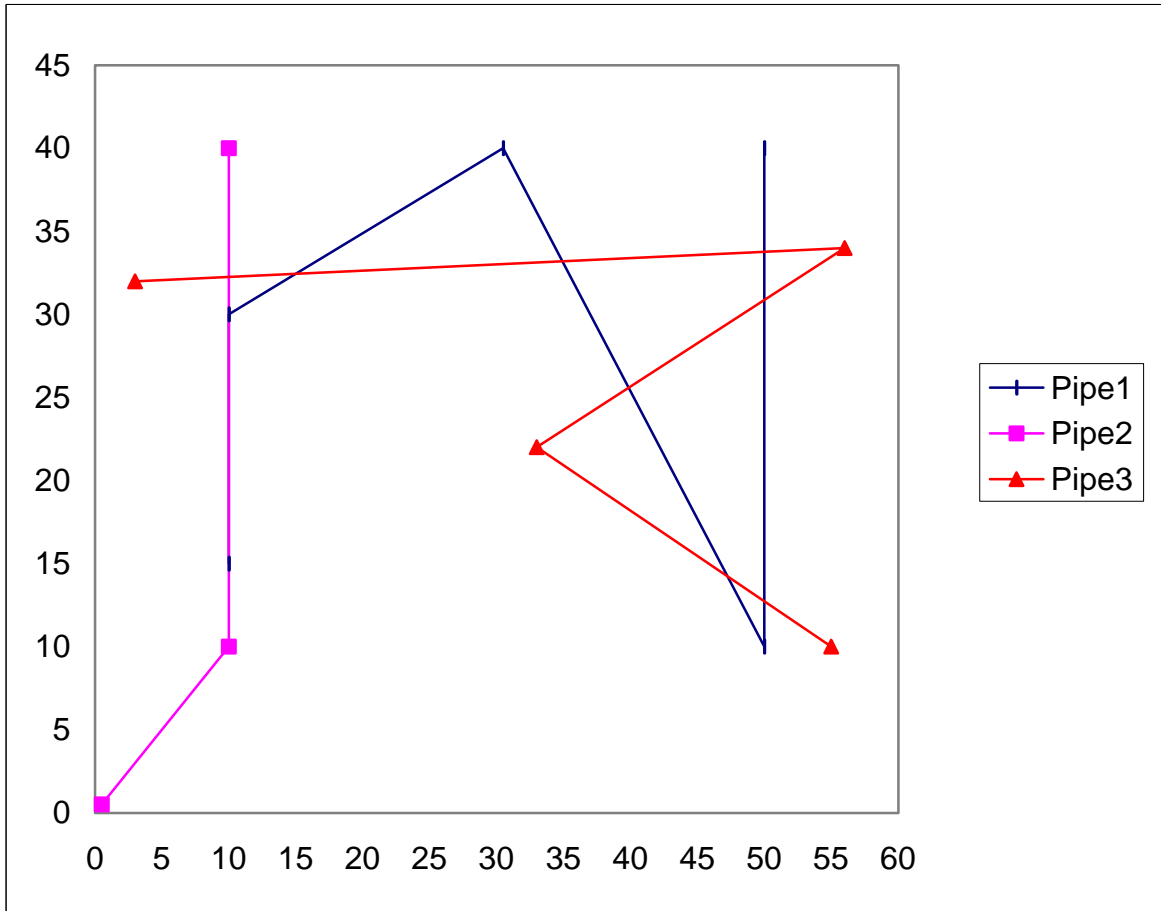
2nd Test

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
10	10	40	10	50	40
20	20	40	20	30	30
40	40	30	10	20	30
50	10	30	20	30	40
		10	20	20	40
				10	30
				20	10



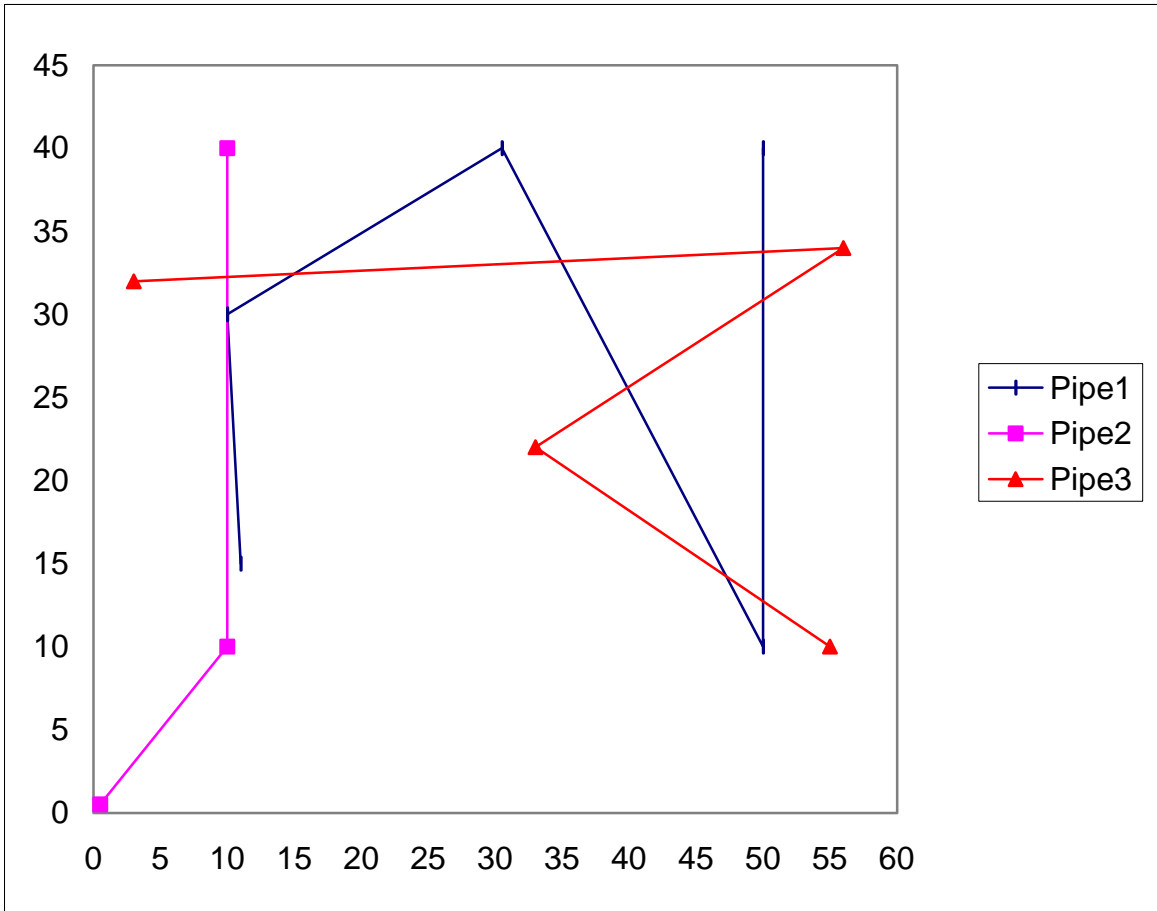
3rd Test

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
10	15	0.5	0.5	55	10
10	30	10	10	33	22
30.5	40	10	40	56	34
50	10			3	32
50	40				



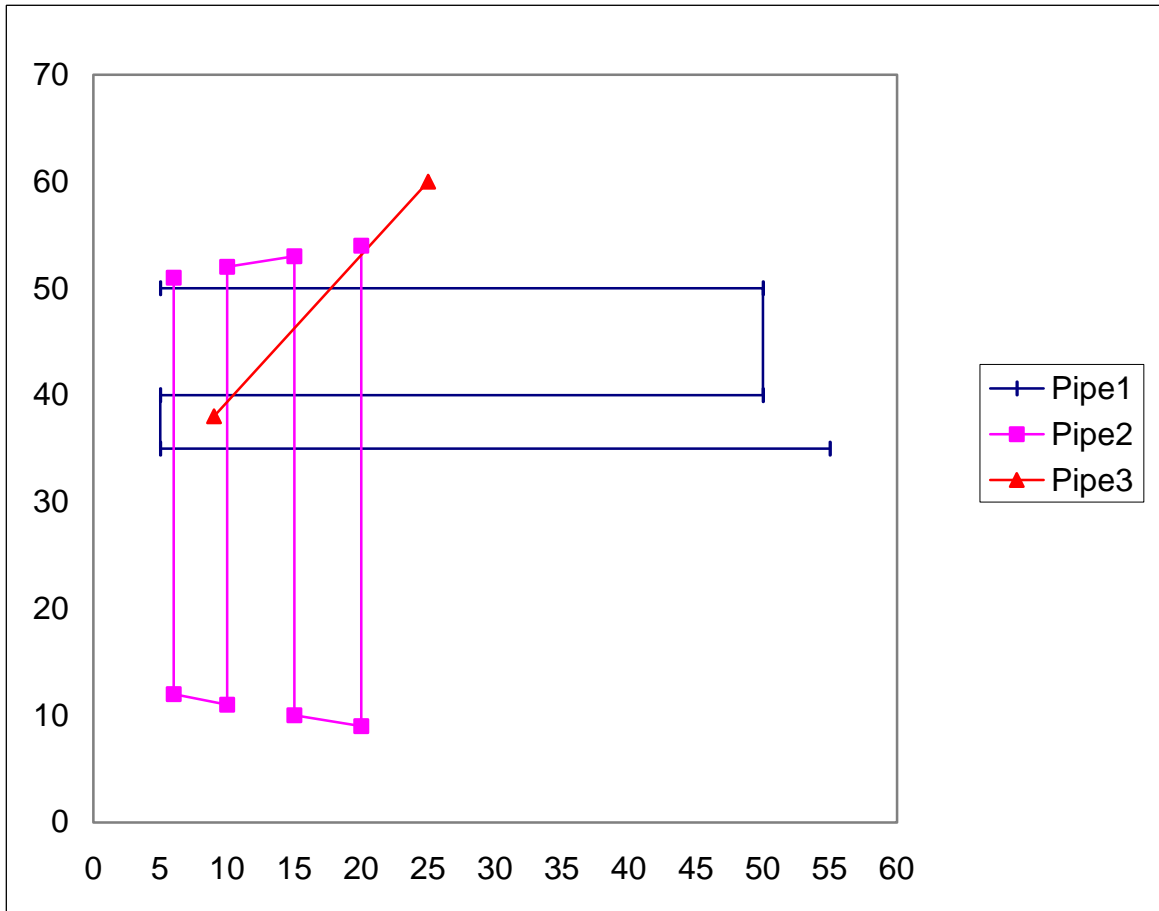
4th Test

Pipe 1 (x) (y)		Pipe 2 (x) (y)		Pipe 3 (x) (y)	
11	15	0.5	0.5	55	10
10	30	10	10	33	22
30.5	40	10	40	56	34
50	10			3	32
50	40				



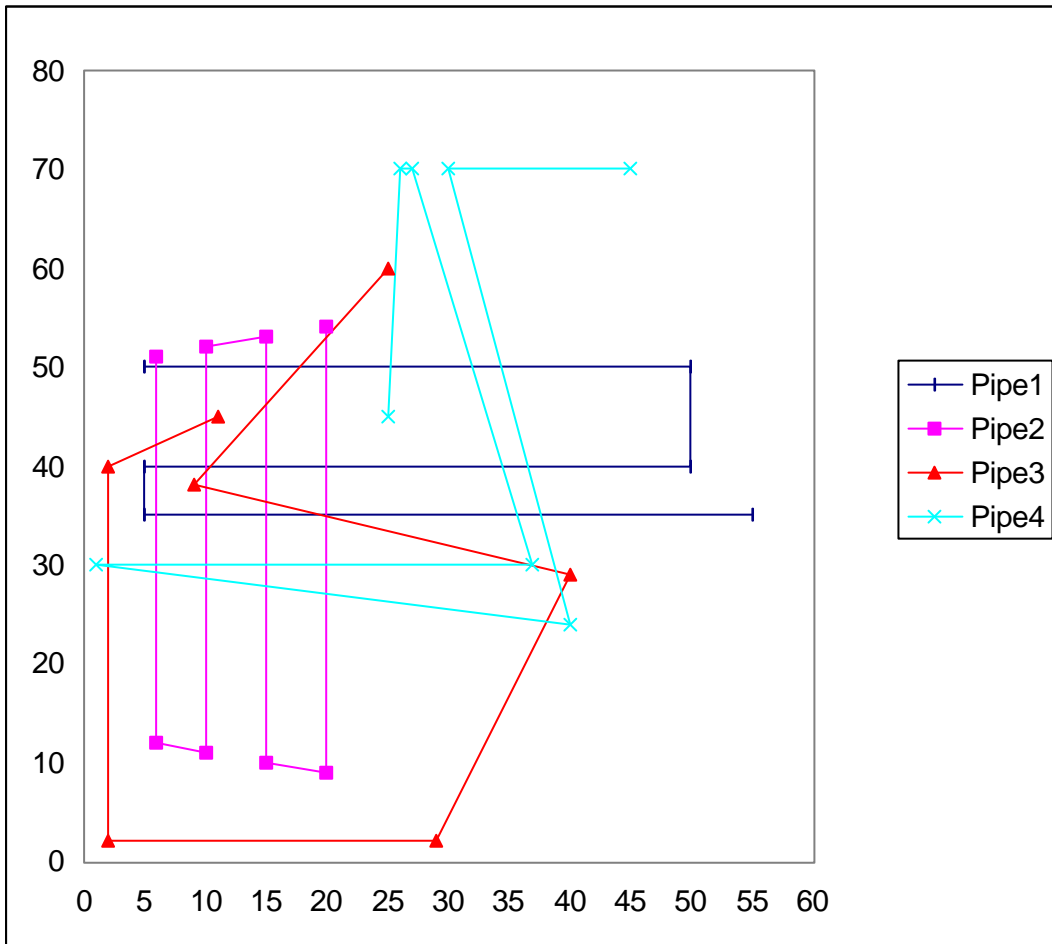
5th Test

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
5	50	6	51	25	60
50	50	6	12	9	38
50	40	10	11		
5	40	10	52		
5	35	15	53		
55	35	15	10		
		20	9		
		20	54		



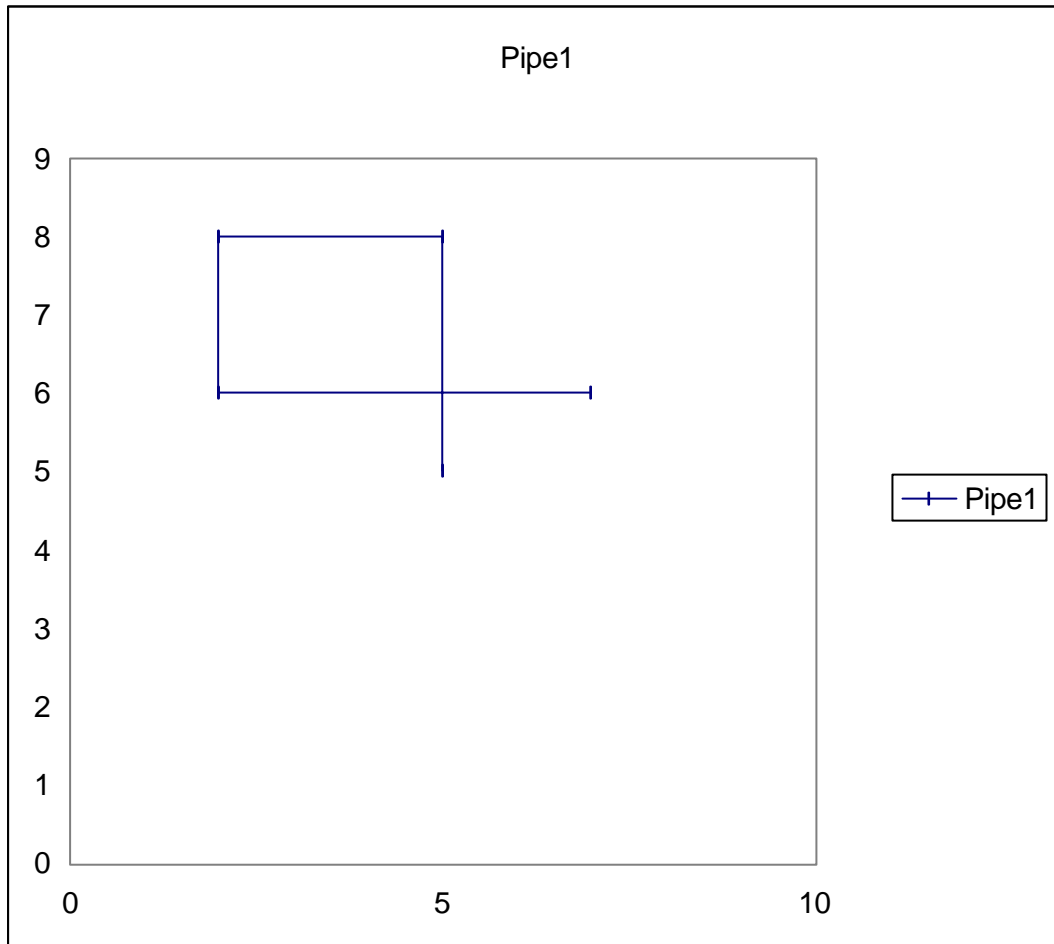
6th Test

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)	Pipe 4 (x)	(y)
5	50	6	51	25	60	45	70
50	50	6	12	9	38	30	70
50	40	10	11	40	29	40	24
5	40	10	52	29	2	1	30
5	35	15	53	2	2	37	30
55	35	15	10	2	40	27	70
		20	9	11	45	26	70
		20	54			25	45



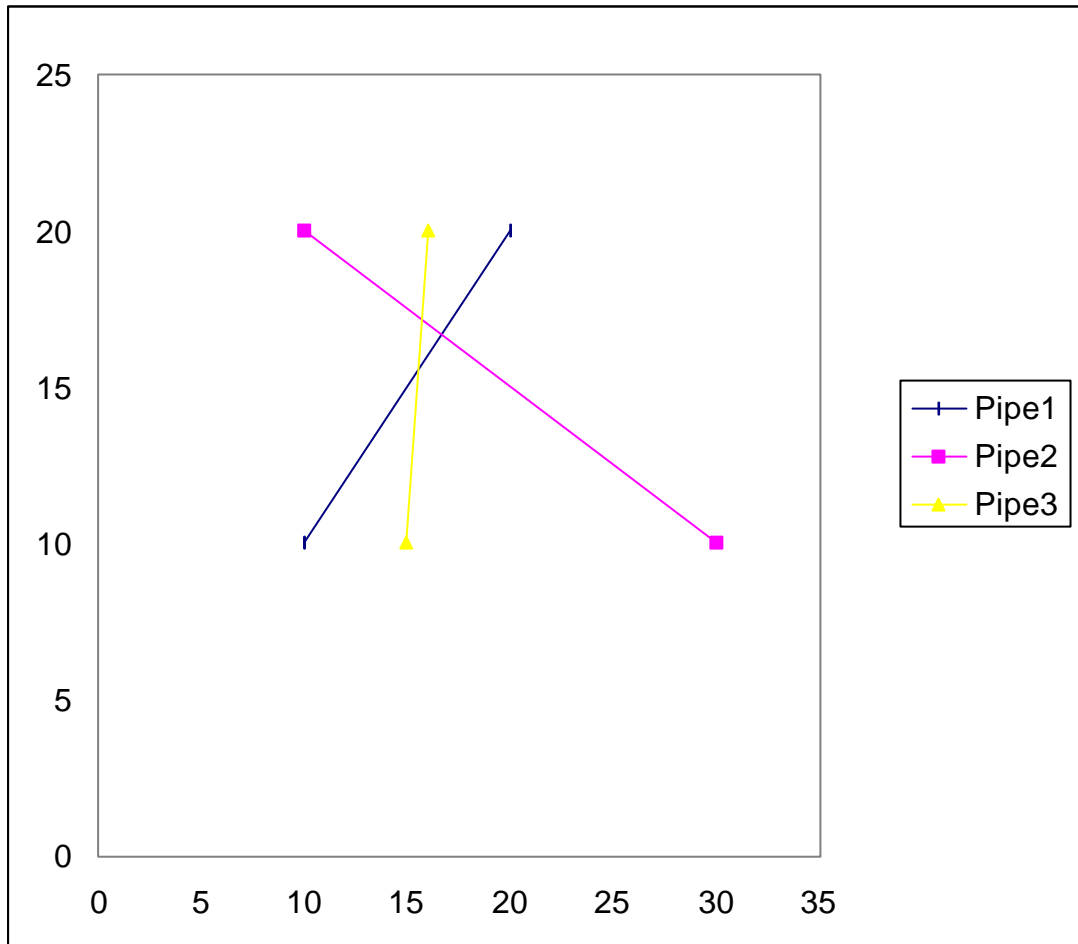
7th Test

Pipe 1 (x)	(y)
5	5
5	8
2	8
2	6
7	6



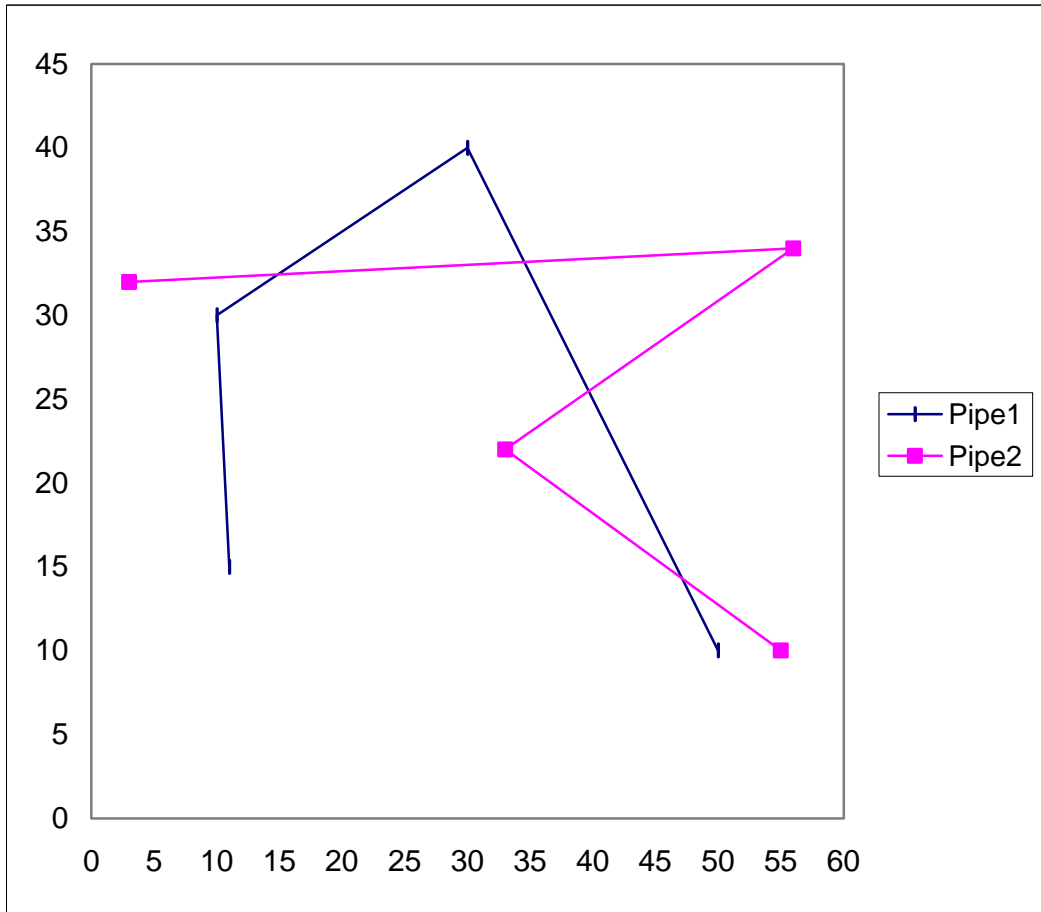
Why 1

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
10	10	10	20	15	10
20	20	30	10	16	20



Why 2

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)
11	15	55	10
10	30	33	22
30	40	56	34
50	10	3	32



Why 3

Pipe 1 (x)	(y)	Pipe 2 (x)	(y)	Pipe 3 (x)	(y)
5	10	5	8	7	8
10	10	7	11	8	11

