Name : Hoseung Lee
ID: 914001975
Email: hoslee@ucdavis.edu
Github repo: https://github.com/mathpop09/ECS-178

How to compile project (must have freeglut on computer):

make

Window Size Setup:

The window is 800 by 800 by default

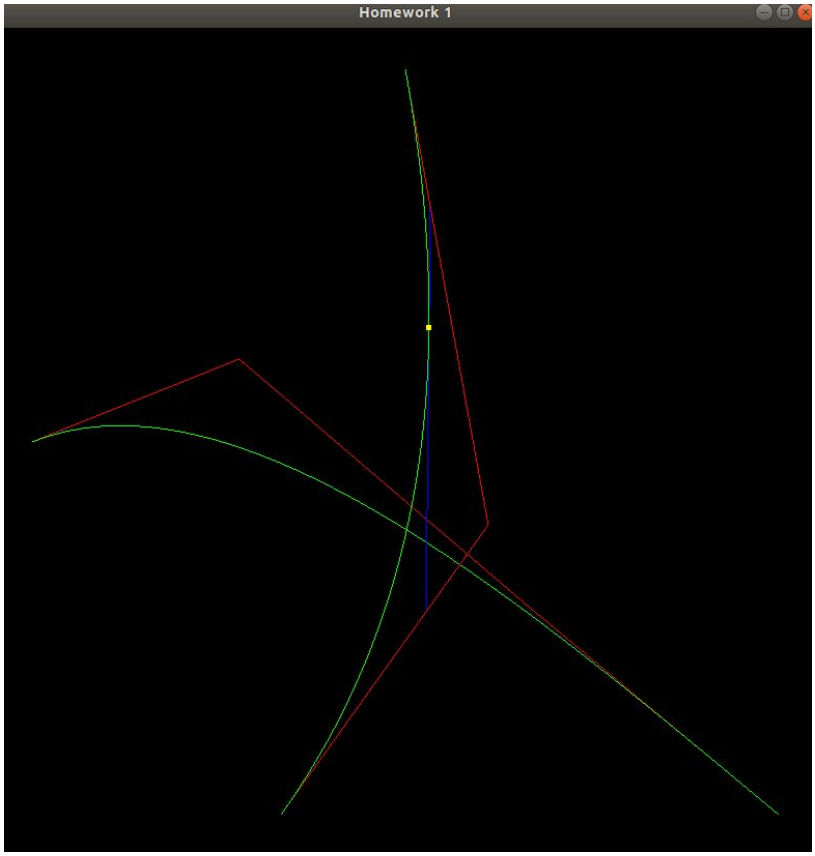
Program Usage:

Input System: The user must input their commands into the command line.

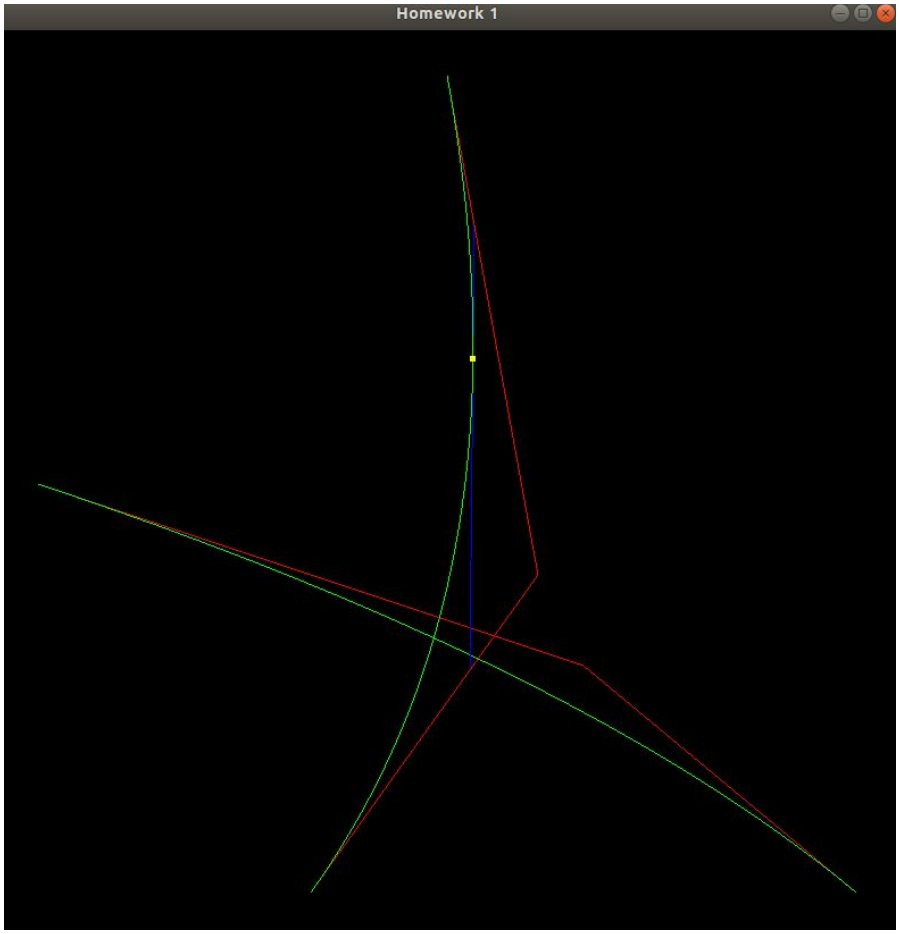
Example of curve creation and deletion:

(Black text is the output, and the highlighted text is the user input. Arrows are used to illustrate the happenings)
./source.out
The dimensions of the window is 800 by 800, the bottom left is (0, 0)
Hello! Would you like to perform an intersection or would you just like to draw some curves? C for Curve Drawing/I for Intersection
C ← The program goes into curve drawing mode
What resolution would you like your curves to be?
40 ← basically 40 lines segments will be used to create the curve
Please tell me how many curves would you like
2 ← two curves will be created
Bernstein method or Casteljau method for drawing curves? [B]ernstein/[C]asteljau:
B ← first curve will use the Bernstein method
Hello! How many coordinates are on your control polygon?
4 ← coordinates on specific control polygon
Please enter your 4 control points. Format: x y
Control Point 1:
-0.9 0
Coordinate X: -0.9 Coordinate Y: 0
Control Point 2:
-0.4 0.2
Coordinate X: -0.4 Coordinate Y: 0.2
Control Point 3:
0.3 -0.4
Coordinate X: 0.3 Coordinate Y: -0.4
Control Point 4:

0.9 -0.9
Coordinate X: 0.9 Coordinate Y: -0.9
Bernstein method or Casteljau method for drawing curves? [B]ernstein/[C]asteljau:
C← second curve will be created using Casteljau method
Hello! How many coordinates are on your control polygon?
3
Please enter your 3 control points. Format: x y
Control Point 1:
0 0.9
Coordinate X: 0 Coordinate Y: 0.9
Control Point 2:
0.2 -0.2
Coordinate X: 0.2 Coordinate Y: -0.2
Control Point 3:
-0.3 -0.9
Coordinate X: -0.3 Coordinate Y: -0.9
Please enter your parameter for t:
0.3 ← this is for the scale (on 0 - 1.0) where a specific point on the curve is drawn
Here are the collection of coordinates:
Here are the collection of Casteljau Curves:
Curve #: 0
  Index: 0 0, 0.9
  Index: 1 0.2, -0.2
  Index: 2 -0.3, -0.9
Here are the collection of Bernstein Curves:
Curve #: 0
  Index: 0 -0.9, 0
  Index: 1 -0.4, 0.2
  Index: 2 0.3, -0.4
  Index: 3 0.9, -0.9



Please send me the curve you want to modify and the proper modifications
[(C)asteljau/(B)ernstein] [Curve Number] [Index Number] [(I)nsert/(D)elete]
B
0
1
D ← Bernstein Curve, Curve # 0, Index 1 will be deleted.

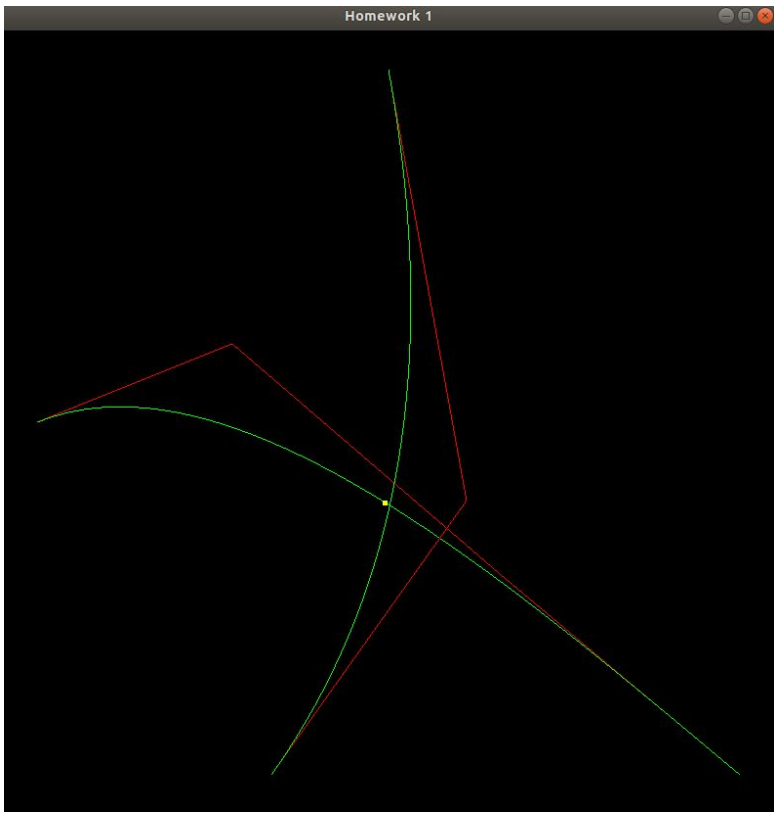


DONE! (Note that you can do this forever as long as you desire)

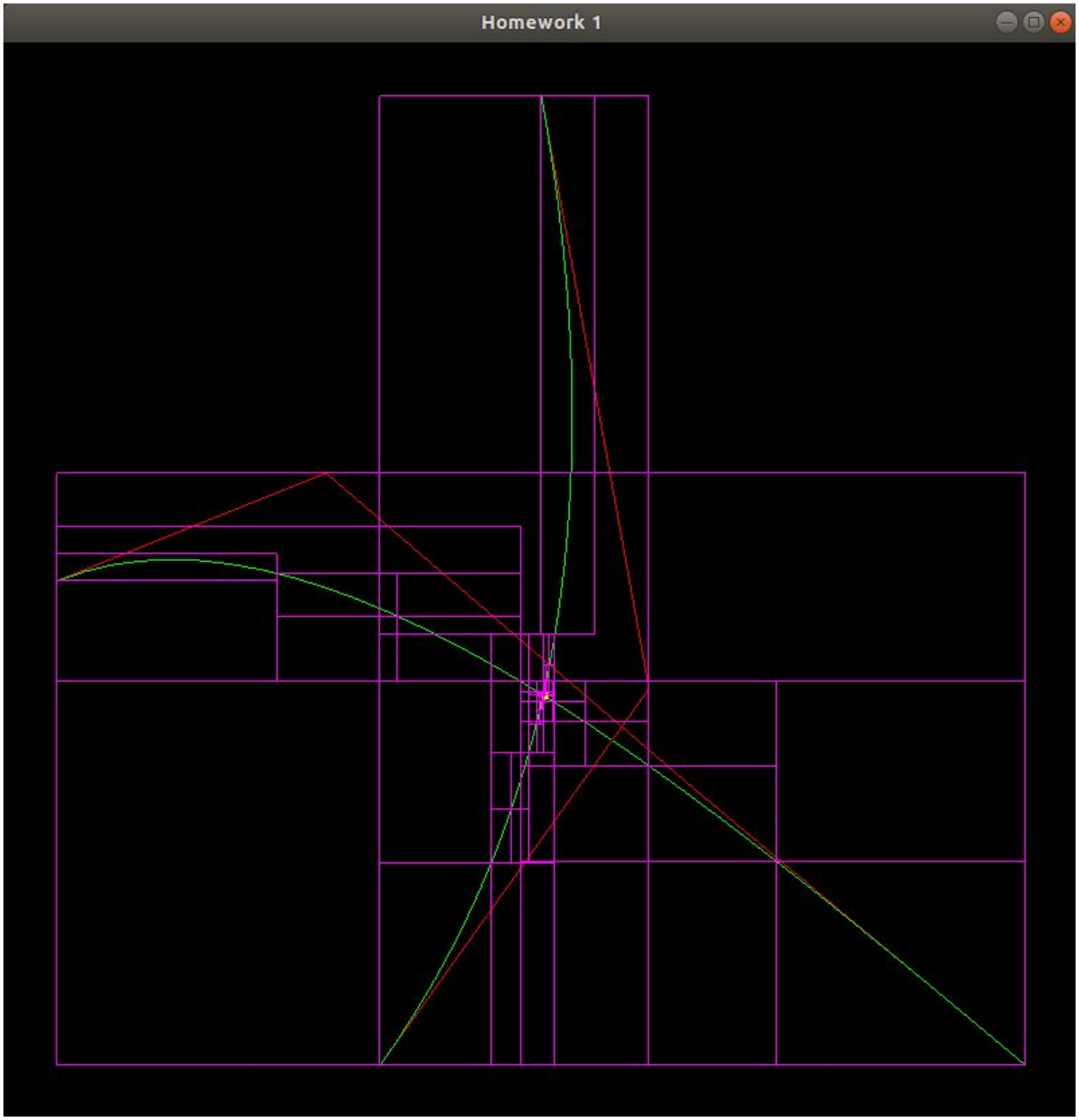
Now for the Intersection Example:

./source.out
The dimensions of the window is 800 by 800, the bottom left is (0, 0)
Hello! Would you like to perform an intersection or would you just like to draw some curves? C for Curve Drawing/I for Intersection
I ← Intersection mode is chosen
What resolution would you like your curves to be?
40 ← same as the curve drawing resolution
Please enter the tolerance that you desire (0.01 or lower recommended)
0.01 ← tolerance for detecting intersection
Please enter the two curves that you desire
Hello! How many coordinates are on your control polygon?
4
Please enter your 4 control points. Format: x y
Control Point 1:
-0.9 0
Coordinate X: -0.9 Coordinate Y: 0

Control Point 2:
-0.4 0.2
Coordinate X: -0.4 Coordinate Y: 0.2
Control Point 3:
0.3 -0.4
Coordinate X: 0.3 Coordinate Y: -0.4
Control Point 4:
0.9 -0.9
Coordinate X: 0.9 Coordinate Y: -0.9
Hello! How many coordinates are on your control polygon?
3
Please enter your 3 control points. Format: x y
Control Point 1:
0 0.9
Coordinate X: 0 Coordinate Y: 0.9
Control Point 2:
0.2 -0.2
Coordinate X: 0.2 Coordinate Y: -0.2
Control Point 3:
-0.3 -0.9
Coordinate X: -0.3 Coordinate Y: -0.9



DONE! An even lower tolerance would show the intersection point more accurately



^Visualization of all the boxes that are used to find the intersection point.

de Casteljau VS Bernstein time-comparison.

I used this code from https://www.pluralsight.com/blog/software-development/how-to-measure-execution-time-intervals-in-c-- to calculate time difference between Bernstein and de Casteljau's

```
#include <chrono>
auto finish = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> elapsed = finish - start;
```

std::cout << "Elapsed time: " << elapsed.count() << " s\n";

Curve tested: resolution: 50, 4 Points: {-0.8 -0.8, 0 0, 0.2 -0.8, 0.6 0}

Bernstein Time: 0.00685875 seconds
Casteljau Time: 0.00990407 seconds

According to this test, Bernstein is 44.4% faster than Casteljau's

Checklist:

A) Casteljau implemented with line segment drawing. Active t-value selection and point insertion + deletion is included in the program.
B) Bernstein implemented. Active point insertion and deletion is possible.
C) Subdivison is implemented. The routine is used in the curve-curve intersection calculations. One thing I left out was prompting for possible subdivision during regular curve drawing routine.
D) Curve-curve intersection implemented using subdivision routine from C).

Other: Can change points and insert points, but cannot do so by actually clicking on the window.