```
#include <stdio.h>
#include <cuda.h>

//Some inline vector math functions
__forceinline__ __device__ float2 operator+(float2 a, float2 b){
  float2 c;
  c.x = a.x + b.x;     c.y = a.y + b.y;
  return c;
}

__forceinline__ __device__ float2 operator-(float2 a, float2 b){
  float2 c;
  c.x = a.x - b.x;     c.y = a.y - b.y;
  return c;
}

__forceinline__ __device__ float2 operator*(float a, float2 b){
  float2 c;
  c.x = a * b.x;     c.y = a * b.y;
  return c;
}

__forceinline__ __device__ float length(float2 a){
  return sqrtf(a.x*a.x + a.y*a.y);
}

#define MAX_TESS_POINTS 32

1. struct BezierLine //A structure containing all the parameters we
     need to tessellate a Bezier line
   {
     float2 CP[3];                          //Control points for the line
     float2 vertexPos[MAX_TESS_POINTS]; //Vertex position array to
              tessellate into
     int nVertices;                        //Number of tessellated
              vertices
   };

   __global__ void computeBezierLines(BezierLine *bLines, int nLines)
   {
     int bidx = blockIdx.x;
     if(bidx < nLines){
     //Compute the curvature of the line
2.   float curvature = length(bLines[bidx].CP[1] - 0.5f*
         (bLines[bidx].CP[0] + bLines[bidx].CP[2]))/length
         (bLines[bidx].CP[2] - bLines[bidx].CP[0]);
     //From the curvature, compute the number of tessellation points
3.   int nTessPoints = min(max((int)(curvature*16.0f),4),32);
4.   bLines[bidx].nVertices = nTessPoints;

     //Loop through the vertices to be tessellated, incrementing by
         blockDim.x
5.    for(int inc = 0; inc < nTessPoints; inc += blockDim.x){
```