```
6.      int idx = inc + threadIdx.x;  //Compute a unique index for
          this point
7.      if(idx < nTessPoints){
8.      float u = (float)idx/(float)(nTessPoints-1);  //Compute u
          from idx
9.      float omu = 1.0f - u;   //pre-compute one minus u

10.     float B3u[3]; //Compute quadratic Bezier coefficients
11.     B3u[0] = omu*omu;
12.     B3u[1] = 2.0f*u*omu;
13.     B3u[2] = u*u;

14.     float2 position = {0,0};  //Set position to zero
15.     for(int i = 0; i < 3; i++){
          //Add the contribution of the i'th control point to position
16.       position = position + B3u[i] * bLines[bidx].CP[i];
        }
        //Assign the value of the vertex position to the correct
        array element
17.     bLines[bidx].vertexPos[idx] = position;
      }
    }
  }
}

18. #define N_LINES 256
19. #define BLOCK_DIM 32

20. int main( int argc, char **argv )
    {
21. BezierLine *bLines_h = new BezierLine[N_LINES]; //Allocate array
        of lines in host memory

    float2 last = {0,0};    //Set initial point to zero (last is the
        last point in the previous segment).
22.  for(int i = 0; i < N_LINES; i++){
23.   bLines_h[i].CP[0] = last; //Set first point of this line to last
        point of previous line
24.    for(int j = 1; j < 3; j++){
       bLines_h[i].CP[j].x = (float)rand()/(float)RAND_MAX;  //Assign
         random coordinate between 0 and 1
       bLines_h[i].CP[j].y = (float)rand()/(float)RAND_MAX;  //Assign
         random coordinate between 0 and 1
       }
       last = bLines_h[i].CP[2];   //Keep the last point of this line
       bLines_h[i].nVertices = 0;  //Set number of tessellated
         vertices to zero
      }

25.   BezierLine *bLines_d; //Pointer to array of Bezier lines in
      device memory
26.   cudaMalloc((void**)&bLines_d, N_LINES*sizeof(BezierLine));
        //Allocate device memory for array of Bezier lines
```