```
__global__ void freeVertexMem(BezierLine *bLines, int nLines)
{
int lidx = threadIdx.x + blockDim.x*blockIdx.x;   //Compute a unique
    index for each Bezier line
10. if(lidx < nLines)
11. cudaFree(bLines[lidx].vertexPos);   //Free the vertex memory for
    this line
}

 #define N_LINES 256
#define BLOCK_DIM 64

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

  float2 last = {0,0};     //Set last point to zero
   for(int i = 0; i < N_LINES; i++){
    bLines_h[i].CP[0] = last;   //Set first point of this line to
          last point of previous line
     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].vertexPos = NULL;   //Set the vertex position array
          to NULL
  bLines_h[i].nVertices = 0;  //Set number of tessellated vertices
          to zero
  }

  BezierLine *bLines_d;   //Pointer to array of Bezier lines in
    device memory
  cudaMalloc((void**)&bLines_d, N_LINES*sizeof(BezierLine));
  cudaMemcpy(bLines_d, bLines_h, N_LINES*sizeof(BezierLine),
    cudaMemcpyHostToDevice);

13. computeBezierLinesCDP<<<ceil((float)N_LINES/(float)BLOCK_DIM),
    BLOCK_DIM>>>(bLines_d, N_LINES);

  //Do something to draw the lines here

14. freeVertexMem<<<ceil((float)N_LINES/(float)BLOCK_DIM),
    BLOCK_DIM>>>(bLines_d, N_LINES);
  cudaFree(bLines_d);   //Free the array of lines in device memory
  delete[] bLines_h;    //Free the array of lines in host memory
}
```