```
 1: #include "DegreeHeapAdjacencyList.h"
 2: #include "VertexVectorAdjacencyList.h"
 3:
 4: DegreeHeapAdjacencyList::DegreeHeapAdjacencyList( )
 5: {
 6: }
 7: DegreeHeapAdjacencyList::~DegreeHeapAdjacencyList( )
 8: {
 9:     delete m_heap;
10: }
11:
12: void DegreeHeapAdjacencyList::Allocate( int nVertex )
13: {
14:     AdjacencyList::Allocate( nVertex );
15:
16:     m_heap = new Heap( nVertex );
17: }
18:
19: int DegreeHeapAdjacencyList::RemoveHighestDegreeVertex( int debug )
20: {
21:     // remove the highest degree vertex from heap
22:     std::pair<int, int> vertex;
23:     m_heap->removeFromHeap( vertex );
24:     int iHighestDegreeVertex = vertex.first;
25:
26:     List * neighbors = m_arrAdjLists[ iHighestDegreeVertex ];
27:
28:     if ( debug >= 2 )
29:     {
30:         fprintf( stderr, "  vertice %d tem %d vizinhos\n",
31:                  iHighestDegreeVertex,
32:                  neighbors->size() );
33:     }
34:
35:     // decrease its neighbor's degree
36:     for ( ListNode * node = neighbors->getFirst();
37:         node != NULL;
38:         node = node->next())
39:     {
40:         int iNeighbor = node->getVertex();
41:
42:         // update this vertex's neighbor's list that this vertex is being removed
43:         //m_arrAdjLists[ iNeighbor ]->erase( iHighestDegreeVertex );
44:
45:         if ( m_heap->HasVertex( iNeighbor ) )
46:         {
47:             // remove edge from this vertex
48:             m_nEdges --;
49:
50:             // decrement degree from neighbor
51:             m_heap->DecrementDegree( iNeighbor );
52:         }
53:     }
54:
55:     // remove edges to neighbors, and let the vertex linger
56:     //neighbors.clear( );
57:
58:     return iHighestDegreeVertex;
59:
60: }
61:
62: void DegreeHeapAdjacencyList::updateData( )
63: {
64:
65:     for( int i = 0; i < m_nVertex; i++ )
66:     {
67:         m_heap->insertOnHeap( i, (int) m_arrAdjLists[i]->size() );
68:     }
69: }
```