```
 1: /*
 2:  * DegreeVectorAdjacencyList.cpp
 3:  *
 4:  *  Created on: Jun 11, 2011
 5:  *      Author: darioandrade
 6:  */
 7:
 8: #include "DegreeVectorAdjacencyList.h"
 9: #include "Heap.h"
10:
11: DegreeVectorAdjacencyList::DegreeVectorAdjacencyList (  )
12: {
13: }
14:
15: DegreeVectorAdjacencyList::~DegreeVectorAdjacencyList ( )
16: {
17:     delete [ ] m_vectorDegrees;
18: }
19:
20: void DegreeVectorAdjacencyList::Allocate( int nVertex )
21: {
22:     AdjacencyList::Allocate( nVertex );
23:
24:     m_vectorDegrees = new int [ nVertex ];
25:
26:     for ( int i = 0; i < nVertex; i++ )
27:     {
28:         m_vectorDegrees[ i ] = 0;
29:     }
30: }
31:
32: void DegreeVectorAdjacencyList::addEdge ( int iVertex, int jVertex, bool bUpdateNeighbor, bool bIncEdge )
33: {
34:     AdjacencyList::addEdge( iVertex, jVertex, bUpdateNeighbor, bIncEdge );
35:
36:     m_vectorDegrees[ iVertex ] ++;
37:
38:     if ( bUpdateNeighbor )
39:     {
40:         m_vectorDegrees[ jVertex ] ++;
41:     }
42: }
43:
44: void DegreeVectorAdjacencyList::DecrementDegree( int iVertex )
45: {
46:     m_vectorDegrees[ iVertex ] --;
47: }
48:
49: void DegreeVectorAdjacencyList::SetDegree( int iVertex, int degree )
50: {
51:     m_vectorDegrees[ iVertex ] = degree;
52: }
53:
54: int DegreeVectorAdjacencyList::GetDegree( int iVertex ) const
55: {
56:     return m_vectorDegrees[ iVertex ];
57: }
58:
59: int DegreeVectorAdjacencyList::GetHighestDegreeVertex( ) const
60: {
61:     int highestDegree = 0;
62:     int iHighestDegreeVertex = -1;
63:
64:     // iterate over the degree vector and find highest degree vector
65:     for ( int i = 0; i < GetSize( ); i++ )
66:     {
67:         // swap if higher
68:         if ( m_vectorDegrees[ i ] > highestDegree )
69:         {
70:             highestDegree = m_vectorDegrees[ i ];
71:             iHighestDegreeVertex = i;
72:         }
73:     }
74:
75:     // if no degree > 0, vertex returned will be -1 (no vertex)
76:     return iHighestDegreeVertex;
77: }
78:
79:
80: int DegreeVectorAdjacencyList::RemoveHighestDegreeVertex( int debug )
81: {
82:     int iHighestDegreeVertex = GetHighestDegreeVertex( );
83:
84:     // find neighbors of this vertex
85:     List * neighbors = m_arrAdjLists[ iHighestDegreeVertex ];
86:
87:     if ( debug >= 2 )
88:     {
89:         fprintf( stderr, "  vertice %d tem %d vizinhos e grau: %d\n",
90:                 iHighestDegreeVertex,
91:                 neighbors->size( ),
92:                 GetDegree( iHighestDegreeVertex ) );
93:     }
94:
```

```
 95:     for ( ListNode * node = neighbors->getFirst();
 96:         node != NULL;
 97:         node = node->next())
 98:     {
 99:         int iNeighbor = node->getVertex();
100:
101:         // update this vertex's neighbor's list that this vertex is being removed
102:         // DATS: Nao eh mais necessario tirar o vertice da lista de adjacencias
103:         // do vizinho, uma vez que o vetor de graus (que de fato Ã© consultado)
104:         // jÃ¡ Ã© decrementado
105:         //m_arrAdjLists[ iNeighbor ]->erase( iHighestDegreeVertex );
106:
107:         // if this neighbor still has edges (it means it has not been removed
108:         // otherwise it must have edges, since it is a neighbor from the highestdegreevertex
109:         if ( m_vectorDegrees[ iNeighbor ] > 0 )
110:         {
111:             // remove edge from this vertex
112:             m_nEdges --;
113:
114:             // decrement degree from neighbor
115:             DecrementDegree( iNeighbor );
116:         }
117:     }
118:
119:     // remove edges to neighbors, and let the vertex linger and ...
120:     //neighbors.clear( );
121:     // reset degree
122:     SetDegree( iHighestDegreeVertex, 0 );
123:
124:     return iHighestDegreeVertex;
125: }
```