```cpp
#include "nodes_LLoLL.h"
#include "cnPtrQueue.h"
#include <iostream>
using namespace std;

namespace CS3358_FA2019_A5P2
{
   void Destroy_cList(CNode*& cListHead)
   {
      int count = 0;
      CNode* cNodePtr = cListHead;
      while (cListHead != 0)
      {
         cListHead = cListHead->link;
         delete cNodePtr;
         cNodePtr = cListHead;
         ++count;
      }
      cout << "Dynamic memory for " << count << " CNodes freed"
           << endl;
   }

   void Destroy_pList(PNode*& pListHead)
   {
      int count = 0;
      PNode* pNodePtr = pListHead;
      while (pListHead != 0)
      {
         pListHead = pListHead->link;
         Destroy_cList(pNodePtr->data);
         delete pNodePtr;
         pNodePtr = pListHead;
         ++count;
      }
      cout << "Dynamic memory for " << count << " PNodes freed"
           << endl;
   }

   // do depth-first traversal and print data
   void ShowAll_DF(PNode* pListHead, ostream& outs)
   {
      while (pListHead != 0)
      {
         CNode* cListHead = pListHead->data;
         while (cListHead != 0)
         {
            outs << cListHead->data << "  ";
            cListHead = cListHead->link;
         }
         pListHead = pListHead->link;
      }
   }

   // do breadth-first (level) traversal and print data
   void ShowAll_BF(PNode* pListHead, ostream& outs)
   {
      // to be implemented (part of assignment)
      if(pListHead == 0)
      {
         return;
      }

      CNode *cursor = 0;
      cnPtrQueue q;

      while(pListHead != 0)
      {
         if(pListHead->data !=0)
         {
```

```
70              q.push(pListHead->data);
71           }
72           pListHead = pListHead->link;
73        }
74
75        while(!q.empty())
76        {
77           cursor = q.front();
78           q.pop();
79           outs << cursor->data << "   ";
80           if(cursor->link != 0)
81           {
82              q.push(cursor->link);
83           }
84        }
85
86     }
87  }
88
```