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

namespace CS3358_FA2021_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)
```

```
    {
       if (pListHead == 0)
       {
         return;
       }
       cnPtrQueue q;
       CNode *cursor = 0;

       while (pListHead != 0)
       {
           if(pListHead->data != 0)
           {
             q.push(pListHead->data);
           }

           pListHead = pListHead->link;
       }
       while (!q.empty())
       {
           cursor = q.front();
           q.pop();
           outs << cursor->data << "  ";

           if (cursor->link != 0)
           {
             q.push(cursor->link);
           }
       }
    }
}
```