```cpp
1   #include "nodes_LLoLL.h"
2   #include "cnPtrQueue.h"
3   #include <iostream>
4   using namespace std;
5
6   namespace CS3358_FA2021_A5P2
7   {
8       void Destroy_cList(CNode*& cListHead)
9       {
10          int count = 0;
11          CNode* cNodePtr = cListHead;
12          while (cListHead != 0)
13          {
14              cListHead = cListHead->link;
15              delete cNodePtr;
16              cNodePtr = cListHead;
17              ++count;
18          }
19          cout << "Dynamic memory for " << count << " CNodes freed"
20              << endl;
21      }
22
23      void Destroy_pList(PNode*& pListHead)
24      {
25          int count = 0;
26          PNode* pNodePtr = pListHead;
27          while (pListHead != 0)
28          {
29              pListHead = pListHead->link;
30              Destroy_cList(pNodePtr->data);
31              delete pNodePtr;
32              pNodePtr = pListHead;
33              ++count;
34          }
35          cout << "Dynamic memory for " << count << " PNodes freed"
36              << endl;
37      }
38
39      // do depth-first traversal and print data
40      void ShowAll_DF(PNode* pListHead, ostream& outs)
41      {
42          while (pListHead != 0)
43          {
44              CNode* cListHead = pListHead->data;
45              while (cListHead != 0)
46              {
47                  outs << cListHead->data << "   ";
48                  cListHead = cListHead->link;
49              }
50              pListHead = pListHead->link;
51          }
52      }
53
54      // do breadth-first (level) traversal and print data
55      void ShowAll_BF(PNode* pListHead, ostream& outs)
56      {
57          if(pListHead != NULL);
58          cnPtrQueue A;
59          CNode* cur = NULL;
60          while(pListHead != NULL){
61              if (pListHead->data != NULL){
62                  A.push(pListHead->data);
63              }
64              pListHead = pListHead->link;
65          }
66          while(!A.empty()){
```

```cpp
        cur = A.front();
        A.pop();
        outs << cur->data << "   ";
        if (cur->link != NULL){
            A.push(cur->link);
        }
    }
}
}
```