05/14/17 09:59:38 /Users/hostname/Desktop/CSE 330/Lab6/[Lab6] Adnar Lozano.cpp

```
 1  // Adnar Lozano
 2  // CSE 330 Data Structures
 3  // Lab 6
 4  // 5/10/17
 5
 6  #include <iostream>
 7  using namespace std;
 8  class Node {
 9  public:
10      int data;
11      Node* next;
12  };
13  Node* Insert(Node* head, int data) {
14      Node* temp = new Node();
15      temp->data = data;
16      temp->next = NULL;
17      if(head != NULL) temp->next = head;
18      head = temp;
19      return head;
20  }
21  void Print(Node* head) {
22      while ( head != NULL) {
23          cout << head->data << " -> ";
24          head = head->next;
25      }
26      cout << "NULL" << endl;
27  }
28  void ReversePrint(Node *head) {
29      if(head == NULL) return;
30      ReversePrint(head->next);
31      cout << head->data << " -> ";
32  }
33  Node* MergeLists(Node* headA, Node* headB) {
34      if (headA == NULL || headB == NULL)
35          return (headA == NULL) ? headB : headA;
36      if (headA->data < headB->data) {
37          headA->next = MergeLists(headA->next, headB);
38          return headA;
39      }
40      headB->next = MergeLists(headB->next, headA);
41      return headB;
42  }
43  int GetNode(Node *head,int positionFromTail) {
44      Node* temp = head;
45      int index = 0;
46      while(head->next) {
47          head = head->next;
48          if(++index>positionFromTail)
49              temp = temp->next;
50      }
51      return temp->data;
52  }
53  bool has_cycle(Node* head) {
54      if (head == NULL) return false;
55      Node* slow = head;
56      Node* fast = head;
57      while (fast && fast->next) {
58          if (fast->next->next == slow)
59              return true;
60          fast = fast->next->next;
61          slow = slow->next;
62      }
63      return false;
64  }
65  int main() {
66      Node* headA = NULL;
67      Node* headB = NULL;
68
```

```
69      headA = Insert(headA, 7);
70      headA = Insert(headA, 5);
71      headA = Insert(headA, 3);
72      headA = Insert(headA, 1);
73
74      headB = Insert(headB, 8);
75      headB = Insert(headB, 6);
76      headB = Insert(headB, 4);
77      headB = Insert(headB, 2);
78
79      cout << "Print List A:\n";
80      Print(headA);
81      cout << "Reversed Print List A:\n";
82      ReversePrint(headA);
83      cout << "NULL\n\n";
84      cout << "Print List B:\n";
85      Print(headB);
86      cout << "Reversed Print List:\n";
87      ReversePrint(headB);
88      cout << "NULL\n\n";
89      cout << "Print Merged List:\n";
90      MergeLists(headA,headB);
91      Print(headA);
92      cout << endl;
93      cout << "PositionFromTail at (2):\n" << GetNode(headA, 2) << "\n\n";
94      cout << "Check if List has a cycle:\n";
95      if (has_cycle(headA) == 1) cout << "List has a cycle\n";
96      else cout<< "List does NOT have a cycle\n\n";
97  }
```