```cpp
#include <iostream>
#include <stdlib.h> //srand, rand
#include <time.h>//clock_t, clock, CLOCKS_PER_SEC
using namespace std;

// reversing a singly linked list


class Node{

    private:
        int data;
        Node* nextNodePtr;

    public:
        Node(){}

        void setData(int d){
            data = d;
        }

        int getData(){
            return data;
        }

        void setNextNodePtr(Node* nodePtr){
                nextNodePtr = nodePtr;
        }

        Node* getNextNodePtr(){
            return nextNodePtr;
        }

};

class List{

    private:
        Node *headPtr;

    public:
        List(){
            headPtr = new Node();
            headPtr->setNextNodePtr(0);
        }

        Node* getHeadPtr(){
            return headPtr;
        }

        bool isEmpty(){

            if (headPtr->getNextNodePtr() == 0)
                return true;

            return false;
        }


        void insert(int data){

            Node* currentNodePtr = headPtr->getNextNodePtr();
            Node* prevNodePtr = headPtr;

```

```
65              while (currentNodePtr != 0){
66                  prevNodePtr = currentNodePtr;
67                  currentNodePtr = currentNodePtr->getNextNodePtr();
68              }
69
70              Node* newNodePtr = new Node();
71              newNodePtr->setData(data);
72              newNodePtr->setNextNodePtr(0);
73              prevNodePtr->setNextNodePtr(newNodePtr);
74
75          }
76
77          void insertAtIndex(int insertIndex, int data){
78
79              Node* currentNodePtr = headPtr->getNextNodePtr();
80              Node* prevNodePtr = headPtr;
81
82              int index = 0;
83
84              while (currentNodePtr != 0){
85
86                  if (index == insertIndex)
87                      break;
88
89                  prevNodePtr = currentNodePtr;
90                  currentNodePtr = currentNodePtr->getNextNodePtr();
91                  index++;
92              }
93
94              Node* newNodePtr = new Node();
95              newNodePtr->setData(data);
96              newNodePtr->setNextNodePtr(currentNodePtr);
97              prevNodePtr->setNextNodePtr(newNodePtr);
98
99          }
100
101
102
103         void IterativePrint(){
104
105             Node* currentNodePtr = headPtr->getNextNodePtr();
106
107             while (currentNodePtr != 0){
108                 cout << currentNodePtr->getData() << " ";
109                 currentNodePtr = currentNodePtr->getNextNodePtr();
110             }
111
112             cout << endl;
113
114         }
115
116
117         void reverseList(){
118
119             Node* currentNodePtr = headPtr->getNextNodePtr();
120             Node* prevNodePtr = 0;
121             Node* nextNodePtr = currentNodePtr;
122
123             while (currentNodePtr != 0){
124
125                 nextNodePtr = currentNodePtr->getNextNodePtr(); // Step 1
126                 currentNodePtr->setNextNodePtr(prevNodePtr); // Step 2
127                 prevNodePtr = currentNodePtr;  // Step 3
128                 currentNodePtr = nextNodePtr;  // Step 4
```

```
129
130                 }
131
132                 headPtr->setNextNodePtr(prevNodePtr);
133
134             }
135
136
137
138
139
140
141     };
142
143     int main(){
144
145         int listSize;
146
147         cout << "Enter the number of elements you want to insert: ";
148         cin >> listSize;
149
150         List integerList; // Create an empty list
151
152         srand(time(NULL));
153
154         int maxValue;
155
156         cout << "Enter the maximum value for an element: ";
157         cin >> maxValue;
158
159
160         for (int i = 0; i < listSize; i++){
161
162             int value = rand() % maxValue;
163
164             integerList.insertAtIndex(i, value);
165         }
166
167         cout << "Contents of the List (before reversal): ";
168         integerList.IterativePrint();
169
170         integerList.reverseList();
171
172         cout << "Contents of the List (after reversal): ";
173         integerList.IterativePrint();
174
175     return 0;
176     }
```

```
Enter the number of elements you want to insert: 10
Enter the maximum value for an element: 50
Contents of the List (before reversal): 30 36 23 48 47 48 27 41 13 45
Contents of the List (after reversal): 45 13 41 27 48 47 48 23 36 30
```