```
//    Advanced Data Structures - Assignment - 1 :
//    Submitted by
//   Vishnu Prasad Vishwanathan
//   793782749
//   vvishwan@syr.edu

// assignmentAds1.cpp : Defines the entry point for the console application//

#include "stdafx.h"
#include <iostream>
using namespace std;

class node {
public:
	int value;
	node * next;
	node() { next = nullptr; }
	node(int k) { value = k; next = nullptr; }
};

class nexted_list {
public:
	int num_nodes;
	node * head;
	nexted_list() { num_nodes = 0; head = nullptr; }
	void make_random_list(int k);                                    //create a nexted
list of

	//k nodes with values in 0 ..99 randomly
	void print(); //Print values of all nodes from head node

	void reverse(); //Reverse the order of nodes of nexted list
                                //void remove_all(int k); //Remove all nodes whose node
values are k
                                //void insert(int k); //create a new node with value k
and insert it to an already sorted list. After the insert, the nexted list is still
sorted.


                                //**********************
                                //Implement the following member functions.

	void bubble_sort(); //Bubble-Sort the nodes, based on ascending order of node
values
	void selection_sort();                                    //void selection_sort();
//Selection-Sort the nodes, based on ascending order of node values
	void insertion_sort(); //Insert - Sort the nodes, based on ascending order of node
values


};


void nexted_list::make_random_list(int k) {
	node * p;
```

```
	for (int i = 0; i < k; i++) {
		p = new node(rand() % 100);
		p->next = head;
		head = p;
		num_nodes++;
	}
}

void nexted_list::print() {
	node * p = head;
	cout << endl;
	while (p != nullptr) {
		cout << p->value << " ";
		p = p->next;
	}
}

void nexted_list::reverse() {
	if (num_nodes <= 1) return;
	node * p1 = head, *p2 = head->next, *p3;
	while (p2 != nullptr) {
		p3 = p2->next;
		p2->next = p1;
		if (p1 == head) p1->next = nullptr;
		p1 = p2;
		p2 = p3;
	}
	head = p1;
}
void nexted_list::bubble_sort() {
	int i = 0;
	node * p1 = head, *p2 = nullptr, *p3 = nullptr;
	bool swap_done = true;

	while (swap_done) {
		p1 = head;
		swap_done = false;
		while ((p1->next) != p3)
		{
			if (p1->value > (p1->next)->value)
			{
				i = p1->value;
				p1->value = p1->next->value;
				p1->next->value = i;

				swap_done = true;
			}
			p1 = p1->next;
		}p3 = p1;
	}
}

void nexted_list::insertion_sort()
{
	int i = 0;
	node * p1 = new node();
	p1->next = head;
	head = p1;
```

```
	node * p2 = new node();
	node * p3 = new node();

	bool swap_done1 = false;
	while (p1->next != NULL)
	{
		p2 = head;
		swap_done1 = false;
		while (p2 != p1)
		{
			if (p2->next->value > p1->next->value)
			{
				p3 = p1->next;
				p1->next = p1->next->next;
				p3->next = p2->next;
				p2->next= p3;

				swap_done1 = true;
				break;
			}
			else
				p2 = p2->next;
		}
		if (!swap_done1) {
			p1 = p1->next;
		}
	}
}
void nexted_list::selection_sort()
{
	int number;
	node *p1, *p2;
	for (p1 = head; p1 != NULL; p1 = p1->next) {
		for (p2 = p1->next; p2 != NULL; p2 = p2->next) {
			if (p1->value > p2->value) {
				number = p1->value;
				p1->value = p2->value;
				p2->value = number;
			}
		}
	}
}
int main() {

	//Some examples of tests for your program are given below
	//During grading, other test cases will also be used
	cout << "Input List";
	cout << "------------------------------------------------------------------";
	nexted_list L1;
	L1.make_random_list(50);
	L1.print();
	cout << "\n------------------------------------------------------------------\n";
	cout << "\nBubble Sort Output\n";
	cout << "\n------------------------------------------------------------------\n";
	L1.bubble_sort();
	L1.print();

	cout << "\nInput List";
```

```
        cout << "\n-----------------------------------------------------------------\n";
        nexted_list L2;
        L2.make_random_list(50);
        L2.print();
        cout << "\n-----------------------------------------------------------------\n";
        cout << "\nSelection Sort Output\n";
        cout << "\n-----------------------------------------------------------------\n";
        L2.selection_sort();
        L2.print();

        cout << "\nInput List\n";
        cout << "\n-----------------------------------------------------------------\n";
        nexted_list L3;
        L3.make_random_list(50);
        L3.print();
        cout << "\n-----------------------------------------------------------------\n";
        cout << "\nInsertion Sort Output\n";
        cout << "\n-----------------------------------------------------------------\n";
        L3.insertion_sort();
        L3.print();

        getchar();
        getchar();
        return 0;
}
```