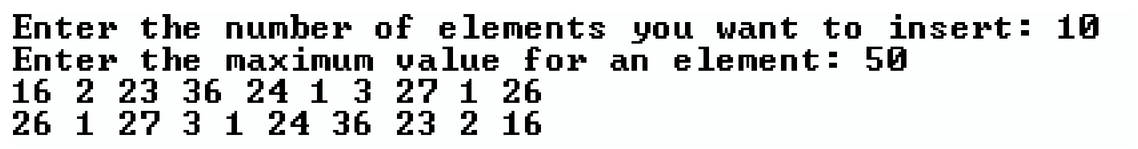
```cpp
#include <iostream>
#include <stdlib.h> //srand, rand
#include <time.h>//clock_t, clock, CLOCKS_PER_SEC
using namespace std;

// implementing a doubly linked list


class Node{

    private:
        int data;
        Node* nextNodePtr;
        Node* prevNodePtr;

    public:
        Node(){}

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

        int getData(){
            return data;
        }

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

        Node* getNextNodePtr(){
            return nextNodePtr;
        }

        void setPrevNodePtr(Node* nodePtr){
                prevNodePtr = nodePtr;
        }

        Node* getPrevNodePtr(){
            return prevNodePtr;
        }

};

class Stack{

    private:
        Node* headPtr;
        Node* tailPtr;


    public:
        Stack(){
            headPtr = new Node();
            tailPtr = new Node();
            headPtr->setNextNodePtr(0);
            tailPtr->setPrevNodePtr(0);
        }

        Node* getHeadPtr(){
            return headPtr;
        }

        Node* getTailPtr(){
            return tailPtr;
        }

        bool isEmpty(){

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

            return false;
        }


        void push(int data){

            Node* newNodePtr = new Node();
            newNodePtr->setData(data);
            newNodePtr->setNextNodePtr(0);

            Node* lastNodePtr = tailPtr->getPrevNodePtr();

            if (lastNodePtr == 0){

                headPtr->setNextNodePtr(newNodePtr);
                newNodePtr->setPrevNodePtr(0);

            }
            else{

                lastNodePtr->setNextNodePtr(newNodePtr);
                newNodePtr->setPrevNodePtr(lastNodePtr);

            }

            tailPtr->setPrevNodePtr(newNodePtr);

        }


        int pop(){

            Node* lastNodePtr = tailPtr->getPrevNodePtr();
            Node* prevNodePtr = 0;

            int poppedData = -100000; //empty stack

            if (lastNodePtr != 0){
                prevNodePtr = lastNodePtr->getPrevNodePtr();
                poppedData = lastNodePtr->getData();
            }
            else
                return poppedData;

            if (prevNodePtr != 0){
                prevNodePtr->setNextNodePtr(0);
                tailPtr->setPrevNodePtr(prevNodePtr);
            }
            else{
                headPtr->setNextNodePtr(0);
                tailPtr->setPrevNodePtr(0);
            }

            return poppedData;

        }
```

```cpp


        int peek(){

            Node* lastNodePtr = tailPtr->getPrevNodePtr();

            if (lastNodePtr != 0)
                return lastNodePtr->getData();
            else
                return -100000; //  empty stack

        }


        void IterativePrint(){

            Node* currentNodePtr = headPtr->getNextNodePtr();

            while (currentNodePtr != 0){
                cout << currentNodePtr->getData() << " ";
                currentNodePtr = currentNodePtr->getNextNodePtr();
            }

            cout << endl;

        }



        void ReversePrint(){

            Node* currentNodePtr = tailPtr->getPrevNodePtr();

            while (currentNodePtr != 0){

                cout << currentNodePtr->getData() << " ";
                currentNodePtr = currentNodePtr->getPrevNodePtr();
            }

            cout << endl;
        }







};

int main(){

    int stackSize;

    cout << "Enter the number of elements you want to insert: ";
    cin >> stackSize;

    Stack stack; // Create an empty stack

    srand(time(NULL));

    int maxValue;

    cout << "Enter the maximum value for an element: ";
    cin >> maxValue;


    for (int i = 0; i < stackSize; i++){

        int value = rand() % maxValue;
        stack.push(value);
        cout << value << " ";
    }

    cout << endl;

    //cout << "Contents of the Stack: ";
    //stack.IterativePrint();


    while (!stack.isEmpty()){

        cout << stack.pop() << " ";
    }

    cout << endl;

return 0;
}
```



```
Enter the number of elements you want to insert: 10
Enter the maximum value for an element: 50
16 2 23 36 24 1 3 27 1 26
26 1 27 3 1 24 36 23 2 16
```