

```

1  #include <iostream>
2  #include <stdlib.h> //srand, rand
3  #include <time.h> //clock_t, clock, CLOCKS_PER_SEC
4  using namespace std;
5
6  // implementing a doubly linked list
7
8
9  class Node{
10
11     private:
12         int data;
13         Node* nextNodePtr;
14         Node* prevNodePtr;
15
16     public:
17         Node() {}
18
19         void setData(int d){
20             data = d;
21         }
22
23         int getData(){
24             return data;
25         }
26
27         void setNextNodePtr(Node* nodePtr){
28             nextNodePtr = nodePtr;
29         }
30
31         Node* getNextNodePtr(){
32             return nextNodePtr;
33         }
34
35         void setPrevNodePtr(Node* nodePtr){
36             prevNodePtr = nodePtr;
37         }
38
39         Node* getPrevNodePtr(){
40             return prevNodePtr;
41         }
42
43 };
44
45 class Stack{
46
47     private:
48         Node* headPtr;
49         Node* tailPtr;
50
51
52     public:
53         Stack(){
54             headPtr = new Node();
55             tailPtr = new Node();
56             headPtr->setNextNodePtr(0);
57             tailPtr->setPrevNodePtr(0);
58         }
59
60         Node* getHeadPtr(){
61             return headPtr;
62         }
63
64         Node* getTailPtr(){

```

```

65     return tailPtr;
66 }
67
68 bool isEmpty() {
69     if (headPtr->getNextNodePtr() == 0)
70         return true;
71     return false;
72 }
73
74 }
75
76 void push(int data) {
77     Node* newNodePtr = new Node();
78     newNodePtr->setData(data);
79     newNodePtr->setNextNodePtr(0);
80
81     Node* lastNodePtr = tailPtr->getPrevNodePtr();
82
83     if (lastNodePtr == 0) {
84         headPtr->setNextNodePtr(newNodePtr);
85         newNodePtr->setPrevNodePtr(0);
86     }
87     else {
88         lastNodePtr->setNextNodePtr(newNodePtr);
89         newNodePtr->setPrevNodePtr(lastNodePtr);
90     }
91     tailPtr->setPrevNodePtr(newNodePtr);
92 }
93
94 int pop() {
95     Node* lastNodePtr = tailPtr->getPrevNodePtr();
96     Node* prevNodePtr = 0;
97
98     int poppedData = -100000; //empty stack
99
100    if (lastNodePtr != 0) {
101        prevNodePtr = lastNodePtr->getPrevNodePtr();
102        poppedData = lastNodePtr->getData();
103    }
104    else
105        return poppedData;
106
107    if (prevNodePtr != 0) {
108        prevNodePtr->setNextNodePtr(0);
109        tailPtr->setPrevNodePtr(prevNodePtr);
110    }
111    else {
112        headPtr->setNextNodePtr(0);
113        tailPtr->setPrevNodePtr(0);
114    }
115
116    return poppedData;
117 }
118
119 }

```

```
129
130
131     int peek(){
132
133         Node* lastNodePtr = tailPtr->getPrevNodePtr();
134
135         if (lastNodePtr != 0)
136             return lastNodePtr->getData();
137         else
138             return -100000; // empty stack
139
140     }
141
142
143     void IterativePrint(){
144
145         Node* currentNodePtr = headPtr->getNextNodePtr();
146
147         while (currentNodePtr != 0){
148             cout << currentNodePtr->getData() << " ";
149             currentNodePtr = currentNodePtr->getNextNodePtr();
150         }
151
152         cout << endl;
153
154     }
155
156
157     void ReversePrint(){
158
159         Node* currentNodePtr = tailPtr->getPrevNodePtr();
160
161         while (currentNodePtr != 0){
162
163             cout << currentNodePtr->getData() << " ";
164             currentNodePtr = currentNodePtr->getPrevNodePtr();
165         }
166
167         cout << endl;
168     }
169
170
171
172
173
174
175
176 };
177
178 int main(){
179
180     int stackSize;
181
182     cout << "Enter the number of elements you want to insert: ";
183     cin >> stackSize;
184
185     Stack stack; // Create an empty stack
186
187     srand(time(NULL));
188
189     int maxValue;
190
191     cout << "Enter the maximum value for an element: ";
192     cin >> maxValue;
```

```
193
194
195     for (int i = 0; i < stackSize; i++){
196
197         int value = rand() % maxValue;
198         stack.push(value);
199         cout << value << " ";
200     }
201
202     cout << endl;
203
204     //cout << "Contents of the Stack: ";
205     //stack.IterativePrint();
206
207
208     while (!stack.isEmpty()){
209
210         cout << stack.pop() << " ";
211     }
212
213     cout << endl;
214
215     return 0;
216 }
```

```
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
```