

STACK

Driver.cpp

```
1.
2.     #include"stack.h"
3.     #include"stack.cpp"
4.
5. int main()
6. {
7.     stack<int>* st = new stack<int>();
8.     if (st->empty())
9.         cout << "Stack is currently empty" << endl;
10.    st->push(1);
11.    st->push(2);
12.    st->push(3);
13.    while (!st->empty())
14.    {
15.        int value = st->pop();
16.        cout << value << endl;
17.    }
18.    system("pause");
19.    return 0;
20. }
```

node.h

```
1. #ifndef NODE_H
2. #define NODE_H
3. #include<iostream>
4. using namespace std;
5.
6. template<class Type>
7. class node
8. {
9. private:
10.     Type data;
11.     node* next;
12. public:
13.     node(Type element = 0);
14.     void setdata(Type pVal);
15.     Type getdata();
16.     node* getnext();
17.     void setnext(node* x);
18. };
19. #endif
20. 
```

stack.h

```
1. #ifndef STACK_H
2. #define STACK_H
3. #include "node.h"
4. #include "node.cpp"
5. template <class Type>
6. class stack
7. {
8. public:
9.     void push(Type element);
10.    Type pop();
11.    bool empty();
12.    stack();
13.
14. private:
15.    Type size;
16.    node<Type>* top;
17. };
18. #endif
19.
```

node.cpp

```
1. #include "node.h"
2.
3. template<class Type>
4. node<Type>::node(Type element)
5. {
6.     data = element;
7.     next = NULL;
8. }
9.
10. template<class Type>
11. void node<Type>::setdata(Type pVal)
12. {
13.     data = pVal;
14. }
15.
16. template<class Type>
17. Type node<Type>::getdata()
18. {
19.     return data;
20. }
21.
22. template<class Type>
23. node<Type>* node<Type>::getnext()
24. {
25.     return next;
26. }
27.
28. template<class Type>
29. void node<Type>::setnext(node* x)
30. {
31.     next = x;
32. }
33.
```

stack.cpp

```
1. #include "stack.h"
2.
3. template <class Type>
4. bool stack<Type>::empty()
5. {
6.     if (top == NULL)
7.         return true;
8.     else
9.         return false;
10. }
11.
12. template <class Type>
13. stack<Type>::stack()
14. {
15.     size = 0;
16.     top = NULL;
17. }
18.
19. template <class Type>
20. void stack<Type>::push(Type element)
21. {
22.     node<Type>* newNode = new node<Type>();
23.     newNode->setdata(element);
24.     → newNode->setnext(top);
25.     top = newNode;
26.     size++;
27. }
28.
29. template <class Type>
30. Type stack<Type>::pop()
31. {
32.     if (!empty())
33.     {
34.         node<Type>* temp = top;
35.         → Type element = temp->getdata();
36.         top = top->getnext();
37.         size--;
38.         delete temp;
39.         return element;
40.     }
41.     else
42.     {
43.         return 0;
44.     }
45. }
46.
```

Output

```
Stack is currently empty
3
2
1
Press any key to continue .
```