STACK

Driver.cpp

```
#include"stack.h"
#include"stack.cpp"

int main()
{
	stack<int>* st = new stack<int>();
	if (st->empty())
		cout << "Stack is currently empty" << endl;
	st->push(1);
	st->push(2);
	st->push(3);
	while (!st->empty())
	{
		int value = st->pop();
		cout << value << endl;
	}
	system("pause");
	return 0;
}

```

node.h

```
#ifndef NODE_H
#define NODE_H
#include<iostream>
using namespace std;

template<class Type>
class node
{
private:
	Type data;
	node* next;
public:
	node(Type element = 0);
	void setdata(Type pVal);
	Type getdata();
	node* getnext();
	void setnext(node* x);
};
#endif

```

stack.h

```cpp
#ifndef STACK_H
#define STACK_H
#include"node.h"
#include"node.cpp"
template <class Type>
class stack
{
public:
    void push(Type element);
    Type pop();
    bool empty();
    stack();

private:
    Type size;
    node<Type>* top;
};
#endif

```

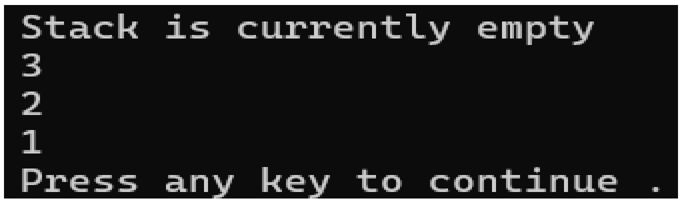
node.cpp

```cpp
#include"node.h"

template<class Type>
node<Type>::node(Type element)
{
      data = element;
      next = NULL;
}

template<class Type>
void node<Type>::setdata(Type pVal)
{
      data = pVal;
}

template<class Type>
Type node<Type>::getdata()
{
      return data;
}

template<class Type>
node<Type>* node<Type>::getnext()
{
      return next;
}

template<class Type>
void node<Type>::setnext(node* x)
{
      next = x;
}

```

stack.cpp

```cpp
#include"stack.h"

template <class Type>
bool stack<Type>::empty()
{
	if (top == NULL)
		return true;
	else
		return false;
}

template <class Type>
stack<Type>::stack()
{
	size = 0;
	top = NULL;
}

template <class Type>
void stack<Type>::push(Type element)
{
	node<Type>* newNode = new node<Type>();
	newNode->setdata(element);
      → newNode->setnext(top);
	top = newNode;
	size++;
}

template <class Type>
Type stack<Type>::pop()
{
	if (!empty())
	{
		node<Type>* temp = top;
	      → Type element = temp->getdata();
		top = top->getnext();
		size--;
		delete temp;
		return element;
	}
	else
	{
		return 0;
	}
}

```

Output



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