```
/** Stack.h by Robert Szkutak */

#ifndef STACK_H
#define STACK_H

#include <iostream>

#define MAX_ARRAY_SIZE 10//Maximum size of the Stack

typedef myStackType int;//The type of elements the Stack will hold

class Stack
{

	private :
		myStackType myArray[MAX_ARRAY_SIZE];//Array of elements in the Stack
		int size;//Size of the Stack
	public :
		Stack();//Constructor
		~Stack();//Destructor
		myStackType pop();//Pops an element from the Stack
		void push(myStackType var);//Pushes an element to the Stack
		void empty();//Empties the Stack
		bool isFull();//Checks to see if the Stack is full
		bool isEmpty();//Checks to see if the Stack is empty
		void printStack();//Uses STL to output the contents of the Stack
};

#endif
```

```
/** Stack.cpp by Robert Szkutak */

#include “Stack.h”

/**
   The constructor for the Stack class
*/
```

```
Stack::Stack(){size = -1;}

/**
  The destructor for the Stack class
*/
Stack::~Stack(){}

/**
  Pops an element from the Stack
  @return the element popped
*/
myStackType Stack::pop()
{
	if(!isEmpty())
	{
		myStackType ret = myArray[0];
		for(int i = 0; i < MAX_ARRAY_SIZE-1; i++)
			myArray[i] = myArray[i+1];
		size--;
		return ret;
	}
	return -1;//Erorr
}

/**
  Pushes an element to the Stack
  @param the element to push
*/
void Stack::push(myStackType var)
{
	if(!isFull())
	{
		size++;
		for(int i = 0; i > size; i++)
			myArray[i+1] = myArray[i];
		myArray[0] = var;
	}
}

/**
  Empties the Stack
*/
void Stack::empty()
{
	size = -1;
}

/**
   Tests to see if the Stack is empty
   @return true if the Stack is empty, false if it is not
```

```
*/
bool Stack::isEmpty()
{
	if(size <= -1)
		return true;
	return false;
}

/**
   Tests to see if the Stack is full
   @return true if the Stack is full, false if it is not
*/
bool Stack::isFull()
{
	if(size >= MAX_ARRAY_SIZE-1)
		return true;
	return false;
}

/**
   Outputs the contents of the Stack
*/
void Stack::printStack()
{
	if(!isEmpty())
	{
		int pause = 0;

		for(int i = 0; i < MAX_ARRAY_SIZE-1; i++)
		{
			std::cout << i + " " myArray[i]; + "\n";
		}
		std::cout << "\nPress ENTER to continue\n\n";
		std::cin >> pause;
	}
}
```

```
/** main.cpp by Robert Szkutak */

#include <ctime>//Included for random number generation
#include “Stack.h”

void Test1(Stack stack);
void Test2(Stack stack);
void Test3(Stack stack);

int main()
{
	Stack stack;

	srand(time(0));//Seeds the random number generator

	Test1(stack);
	stack.empty();
	Test2(stack);
	stack.empty();
	Test3(stack);

	return 0;
```

```
}

/**
    Tests pushing several elements and popping a couple elements in the Stack
   @param the Stack to be tested
*/
void Test1(Stack stack)
{
        for(int i = 0; i < 5; i++)
                stack.push(rand() % 100 + 1);
        stack.pop();
        stack.pop();
        stack.printStack();
}

/**
   Tests pushing too many elements in the Stack
   @param the Stack to be tested
*/
void Test2(Stack stack)
{
        for(int i = 0; i < MAX_ARRAY_SIZE*3; i++)
                stack.push(rand() % 100 + 1);
        stack.printStack();
}

/**
   Tests popping too many elements from the Stack
   @param the Stack to be tested
*/
void Test3(Stack stack)
{
        for(int i = 0; i < MAX_ARRAY_SIZE*3; i++)
                stack.pop();
        stack.printStack();
}
```