#include<iostream>

#include<conio.h>

#include<cassert>

using namespace std;

template <class Type>

class stackType

{

private:

int maxStackSize;

int stackTop;

Type\* list;

void copyStack(const stackType<Type>& otherStack)

{

delete[] list;

maxStackSize = otherStack.maxStackSize;

stackTop = otherStack.stackTop;

list = new Type[maxStackSize];

for (int j = 0; j < stackTop; j++)

list[j] = otherStack.list[j];

}

public:

const stackType<Type>& operator=(const stackType<Type>& otherStack)

{

if (this != &otherStack)

copyStack(otherStack);

return \*this;

}

void initializeStack()

{

stackTop = 0;

}

void destroyStack()

{

stackTop = 0;

}

bool isEmptyStack() const

{

return(stackTop == 0);

}

bool isFullStack() const

{

return(stackTop == maxStackSize);

}

void push(const Type& newItem)

{

if (!isFullStack())

{

list[stackTop] = newItem;

stackTop++;

}

else

cout << "Cannot add to a full stack." << endl;

}

Type top() const

{

assert(stackTop != 0);

return list[stackTop - 1];

}

void pop()

{

if (!isEmptyStack())

stackTop--;

else

cout << "Cannot remove from an empty stack." << endl;

}

void print() const

{

for (int i = 0; i < stackTop; i++)

{

cout << list[stackTop - 1 - i] << " ";

}

}

stackType(int stackSize = 100)

{

if (stackSize <= 0)

{

cout << "Size of the array to hold the stack must be positive." << endl;

cout << "Creating an array of size 100." << endl;

maxStackSize = 100;

}

else

maxStackSize = stackSize;

stackTop = 0;

list = new Type[maxStackSize];

}

stackType(const stackType<Type>& otherStack)

{

list = NULL;

copyStack(otherStack);

}

~stackType()

{

delete[] list;

}

};

void main()

{

stackType<int> stack(10);

stack.initializeStack();

stack.push(45);

int value;

for (int j = 0; j < 5; j++)

{

cout << " Enter Value to be inserted "; cin >> value;

stack.push(value);

}

cout << " The enteries in the Stack are ";

stack.print();

cout << endl;

cout << " The Top value in the Stack is ";

cout << stack.top() << endl;

stack.pop();

cout << " The enteries in the Stack are ";

stack.print();

cout << endl;

\_getch();

}