```
#ifndef STACK_H
#define STACK_H
// Stack.h -- a stack implemented as an adapter (of vector or list or ...)
#include <list>
using namespace std;
//Use the following line for STL containers.
template <class T, template <class T, class = allocator<T> > class Container =
list>
//template <class T, template <class T> class Container = list>
class Stack
{
public:
//We don't need a constructor or destructor because the Container has/should have
one
//Stack(): container() { }
//~Stack() { ~container(); }
bool empty() const { return c.empty(); }
unsigned int size() const { return c.size(); }
void push(const T & x) { c.push_back(x); }
void pop() { c.pop_back(); }
T & top() { return c.back(); }
private:
Container<T> c;
};
#endif


//queue.h  defines the queue class

#ifndef QUEUE_H
#define QUEUE_H

#include<list>
using namespace std;

template <class T, template <class T, class = allocator<T> > class Container =
list>
class queue {
 public:
//like stack the queue derives constructor and destructor from list
    //operations
   bool     empty () { return c.empty(); }
   unsigned int   size  () { return c.size(); }
   T &            front () { return c.front(); }
   T &            back  () { return c.back(); }
   void           push  (T x) { c.push_back(x); }
   void           pop   () { c.pop_front(); }

 protected:
      Container<T> c;
};
#endif

//queuetest.cpp to test queue

#include <iostream>
#include <cassert>
#include "queue.h"
```

```
#include <list>
#include <string>

using namespace std;

int main()
{
  queue<int> q1;
  assert(q1.size() == 0);
  assert(q1.empty());

  q1.push(0);
  q1.push(4);
  q1.push(5);
  q1.push(1);
  assert(q1.size() == 4);
  assert(q1.front() == 0);
  assert(q1.back() == 1);

  q1.push(2);
  assert(q1.size() == 5);
  assert(q1.back() == 2);

  q1.pop();
  assert(q1.front() == 4);
  assert(q1.back() == 2);


  queue<string> q2;
  q2.push("Would");
  q2.push("you");
  q2.push("kindly?");
  assert(q2.front() == "Would");
  assert(q2.back() == "kindly?");

  cout << "All tests passed";

}
// Stack_test.cpp

#include <iostream>
#include <cassert>
#include "stack.h"
#include <string>
#include <vector>
#include <list>
using namespace std;


int main()
{
Stack<int, vector> s1;
assert(s1.size() == 0);
assert(s1.empty());
s1.push(16);
assert(s1.size() == 1);
assert(s1.top() == 16);
s1.pop();
assert(s1.size() == 0);
```

```
s1.push(11);
assert(s1.size() == 1);
assert(s1.top() == 11);
s1.push(22);
assert(s1.size() == 2);
assert(s1.top() == 22);
s1.push(33);
assert(s1.size() == 3);
assert(s1.top() == 33);
s1.pop();
assert(s1.size() == 2);
assert(s1.top() == 22);
Stack<string, list> s2;
s2.push("abc");
s2.push("de");
s2.pop();
assert(s2.top() == "abc");
cout << "SUCCESS\n";
}
// test.cpp - a simple test program for Stack.h
#include <iostream>
#include <vector>
#include "stack.h"
using namespace std;
main()
{
Stack<int> s; // uses List as the default container
s.push(5);
s.push(6);
cout << s.top() << endl;
Stack<double, vector> v; // uses Vector as the container
v.push(1.5);
v.push(2.3);
v.pop();
cout << v.top() << endl;
}

Script started on Wed 07 May 2014 10:42:06 AM PDT
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05##[?
1034h[004470530@jb358-15 lab05]$ g++ -o q_test.##[K queuetest.s##[Kcpp
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15
lab05]$ ./q_test
All tests passed#]0;004470530@jb358-
15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15 lab05]$ G##[Kg-##[K++
-o s_test test.cpp
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15
lab05]$ g++ -o s_test test.cpp######################[K./s_test
6
1.5
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15
lab05]$ g++ -o stack_test stea##[K##[Kack_test.c[[##[K##[Kpp
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15
lab05]$ ./stack_test
SUCCESS
#]0;004470530@jb358-15:/students/csci/004470530/cse330/lab05#[004470530@jb358-15
lab05]$ exit
exit

Script done on Wed 07 May 2014 10:43:47 AM PDT
```