```
/*---------------------First Attempt Array Stacks--------------*/
///////////////////////////////////////////////Vstack.h///////////
#ifndef TONYSTACK
      #define TONYSTACK 1
template<class T>
    class Gstack {
    public:
      virtual void push(T) = 0;     // pure virtual function
      virtual T pop() = 0;          // pure virtual function
      virtual bool isempty() = 0;   // pure virtual function
      virtual bool isfull() = 0;    // pure virtual function
    };
#endif
////////////////////////////////////////////////AStack.h//////////
//#include "Vstack.h"
#ifndef TONYASTACK
      #define TONYASTACK 1
template<class T>
    class Array_stack : public Gstack<T> {
      int sz;
      int top;
      T* astack;
    public:
      Array_stack(int );
      Array_stack();
      ~Array_stack();
            void push(T );
      T pop();
      bool isempty();
      bool isfull();
    };
#endif
///////////////////////////////////////////////LStack.h///////////
//#include "Vstack.h"
#ifndef TONYLSTACK
      #define TONYLSTACK 1
template<class T>
    class Link_stack : public Gstack<T> {
      struct node {
            T data;
            node* next;
            node(T d, node* n)
                  { data = d; next = n; }
      };
      node* top;
    public:
      Link_stack( );
      void push(T );
      T pop();
      bool isempty() ;
      bool isfull() ;
    };
#endif
```

```
/////////////////////////////////////////////ArrayStack.cc/////////
#include <assert.h>
//#include "Astack.h"

template<class T>
      Array_stack<T> :: Array_stack (int Nitems)
            { sz = Nitems; astack = new T[sz]; top = sz; };
template<class T>
      Array_stack<T> :: Array_stack ()
            { sz = 100; astack = new T[sz]; top = sz; };
template<class T>
      Array_stack<T> :: ~Array_stack ()
            { delete[] astack; }
template<class T>
      void Array_stack<T> :: push (T x)
                { assert ( !isfull() ); astack[--top] = x; }
template<class T>
      T Array_stack<T> :: pop()
            { assert( !isempty() ); return( astack[top++] ); }
template<class T>
      bool Array_stack<T> :: isempty()
            { return( top == sz ); }
template<class T>
      bool Array_stack<T> :: isfull ()
            { return ( top == 0 ); }
///////////////////////////////////////////////LinkStack.cc//////////
#include <assert.h>
//#include "Lstack.h"

template<class T>
      Link_stack<T> :: Link_stack(  )
            { top = NULL; }
template<class T>
      void Link_stack<T> :: push(T x)
            { top = new node(x, top); }
template<class T>
      T Link_stack<T> :: pop() {
                  assert( !isempty() );
                  T t = top -> data;
                  node* oldtop = top;
                  top = top -> next;
                  delete oldtop;
                  return t;
      }
template<class T>
      bool Link_stack<T> :: isempty()
            { return( top == NULL ); }
template<class T>
      bool Link_stack<T> :: isfull()
            { return( FALSE ); }
```

```
//////////////////////////////////////////////////Reverse.cc/////////
#include <iostream>
//#include "Vstack.h"
// Doesn't know about Arraystack or Linkedstack
void reverse( Gstack<int>&, int );  // a prototype

void reverse( Gstack<int> &s, int n )
{
        int i;

        for( i = 0; i < n; i++ ) {
                s.push(i);
        }
        for( i = 0; i < n; i++ ) {
                cout << s.pop() << ' ';
        }
        cout << endl;
}

/////////////////////////////////////////     This is Main.cc//////////
//#include "Vstack.h"
//#include "Astack.h"
//#include "Lstack.h"
//extern void reverse( Gstack<int> &s, int n );

int main()
{
      Array_stack<int> s1(500);
      Link_stack<int> s2;     // but not s2();
//    Array_stack<char> s3;
//    Link_stack<char> s4;

      reverse (s1, 20);
      reverse (s2, 20);
//    reverse (s3, 20); //problem reverse only knows about integers
//    reverse (s4, 20); //need to make reverse a member function, perhaps?
      return 0;
}
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*/
```

```
/*----------------------Second Attempt  add linked stacks----------*/
template<class T>
    class Gstack {
    public:
      virtual void push(T) = 0;      // pure virtual function
      virtual T pop() = 0;           // pure virtual function
      virtual bool isempty() = 0;    // pure virtual function
      virtual bool isfull() = 0;     // pure virtual function
    };

template<class T>
    class Array_stack : public Gstack<T> {
      int sz;
      int top;
      T* astack;
    public:
      Array_stack(int Nitems = 100)
            { sz = Nitems; astack = new T[sz]; top = sz; }
      ~Array_stack()
                { delete[] astack; }
      void push(T x)
                { assert( !isfull() ); astack[--top] = x; }
      T pop()
                { assert( !isempty() ); return( astack[top++] ); }
      bool isempty()
                { return( top == sz ); }
      bool isfull()
                { return( top == 0 ); }
    };

template<class T>
    class Link_stack : public Gstack<T> {
      struct node {
            T data;
            node* next;
            node(T d, node* n)
                      { data = d; next = n; }
      };
      node* top;
    public:
      Link_stack( int Nitems = 100 )
                { top = NULL; }
      void push(T x)
                { top = new node(x, top); }
      T pop() {
            assert( !isempty() );
            T t = top -> data;
            node* oldtop = top;
            top = top -> next;
            delete oldtop;
            return t;
      }
```

```
      bool isempty()
                { return ( top == NULL ); }
      bool isfull()
                { return ( FALSE ); }
    };

#include <iostream>
//#include "Vstack.h"
// Doesn't know about Arraystack or Linkedstack
//void reverse( Gstack<int> &, int );      // a prototype

template <class T>
void reverse( Gstack<T> &s, int n )
{
        int i;

        for( i = 0; i < n; i++ ) {
                s.push((T)('a'+i));
        }
        for( i = 0; i < n; i++ ) {
                cout << s.pop() << ' ';
        }
        cout << endl;
}
//#include "Vstack.h"
//#include "Linkedstack.h"
//#include "Arraystack.h"
//extern void reverse( stack<int> &s, int n );

int main()
{
      Array_stack<int> s1(500);
      Link_stack<int> s2;             // or s2(137); but not s2();
      Array_stack<char> s3;
      Link_stack<char> s4(50);

      reverse (s1, 20);
      reverse (s2, 20);
      reverse (s3, 20); //problem reverse only knows about integers
      reverse (s4, 20); //need to make reverse a member function, perhaps?
      return 0;
}
```

```
/*
//////////////Code Warrior 4 says:////////////////
Warning : variable / argument 'Nitems' is not used in function
 (instantiating: 'Link_stack<int>::Link_stack(int)')
TStack.cc line 49   { top = NULL; }

Warning : variable / argument 'Nitems' is not used in function
 (instantiating: 'Link_stack<char>::Link_stack(int)')
TStack.cc line 49   { top = NULL; }

116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
t s r q p o n m l k j i h g f e d c b a
t s r q p o n m l k j i h g f e d c b a
*/
/*
/////////////g++ says//////////////
TStack.cc:92: warning: `class Array_stack<int>' has virtual functions
but non-virtual destructor
TStack.cc:94: warning: `class Array_stack<char>' has virtual functions
but non-virtual destructor

116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
t s r q p o n m l k j i h g f e d c b a
t s r q p o n m l k j i h g f e d c b a
*/
```

```
/*---------------Final attempt Virtual Stacks ------------------*/
template<class T>
    class Gstack {
    public:
      virtual void push(T) = 0;      // pure virtual function
      virtual T pop() = 0;           // pure virtual function
      virtual bool isempty() = 0;    // pure virtual function
      virtual bool isfull() = 0;     // pure virtual function
            void reverse ( int);           // a prototype only
    };
template<class T>
    class Array_stack : public Gstack<T> {
      int sz;
      int top;
      T* astack;
    public:
      Array_stack(int Nitems )
            { sz = Nitems; astack = new T[sz]; top = sz; }
      Array_stack( )
            { sz = 100; astack = new T[sz]; top = sz; }
      ~Array_stack()
                { delete[] astack; }
      void push(T x)
                { assert( !isfull() ); astack[--top] = x; }
      T pop()
                { assert( !isempty() ); return( astack[top++] ); }
      bool isempty()
                { return( top == sz ); }
      bool isfull()
                { return( top == 0 ); }
    };
template<class T>
    class Link_stack : public Gstack<T> {
      struct node {
            T data;
            node* next;
            node(T d, node* n)
                      { data = d; next = n; }
      };
      node* top;
    public:
      Link_stack( )
                { top = NULL; }
      void push(T x)
                { top = new node(x, top); }
      T pop() {
                  assert( !isempty() );
                  T t = top -> data;
                  node* oldtop = top;
                  top = top -> next;
                  delete oldtop;
                  return t;
      }
```

```
      bool isempty()
                { return( top == NULL ); }
      bool isfull()
                { return( FALSE ); }
    };

template <class T>
    void Gstack <T> :: reverse( int n )
    {
        int i;

        for( i = 0; i < n; i++ ) {
                this->push((T)('a'+i));
        }
        for( i = 0; i < n; i++ ) {
                cout << this->pop() << ' ';
        }
        cout << endl;
    }

int main()
{
      Array_stack<int> s1(500);
      Link_stack<int> s2;           // but not s2();
      Array_stack<char> s3;
      Link_stack<char> s4;

      s1.reverse (20);
      s2.reverse (20);
      s3.reverse (20);  //member function reverse knows about integers and chars
      s4.reverse (20);  //as well as arrays and linked lists.
      return 0;
}
/*
////////////Code Warrior //////////

116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
t s r q p o n m l k j i h g f e d c b a
t s r q p o n m l k j i h g f e d c b a

/////////////g++ -Wall -ansi //////////////
VStack.cc:93: warning: `class Array_stack<int>' has virtual functions but
non-virtual destructor
VStack.cc:95: warning: `class Array_stack<char>' has virtual functions but
non-virtual destructor

116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
116 115 114 113 112 111 110 109 108 107 106 105 104 103 102 101 100 99 98 97
t s r q p o n m l k j i h g f e d c b a
t s r q p o n m l k j i h g f e d c b a
*/
```