```
#include <assert.h>
#include <iostream.h>
//////////////////////mydefs1a.h//////////////
class Istack {
      int sz;
      int top;
      int* stack;
public:
      // Note all necessary prototypes are provided inline
      Istack (int nitems = 100)
          { sz = nitems; stack = new int[sz]; top = sz; }
      void push(int x)
          { assert( !isfull() ); stack[--top] = x; }
      int pop()
          { assert( !isempty() ); return( stack[top++] ); }
      bool isempty()
          { return( top == sz ); }
      bool isfull()
          { return( top == 0); }
};
//////////////////stackmain1a.cc////////////
int main()
{
      Istack s1(500);
      Istack s2;              // default constructor?
      Istack s3();            // a function which returns Istack
      Istack s6(137);
      Istack s4[5];           // default constructor called 5 times here
      Istack* s5;             // no space, just a pointer
//    Istack s8[2](137);      // array of two stacks size 137, NO
//    Istack s7()[2];         // s7() is a function returning an array!
//    Istack s9(23)[2];       // two 23-element stacks?
                              // No way, can only use default
      int i;

      for( i = 0; i < 20; i++ ) {
           s1.push(i);             // s1 is a 500 element stack
      }

      for( i = 0; i < 20; i++ ) {
           cout << s1.pop() << ' ';
      }
      cout << endl;


      s4[3].push(11);             // 3rd stack in the array
      cout << s4[3].pop() << ' ';

      s5 = new Istack;            // get stack space, which constructor?
      s5->push(12);               // pointing to a stack
      cout << s5->pop() << endl;
      delete s5;                  // explicit removal of stack
}
```



```
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
11 12
*/
//--------------------------------------
#include <assert.h>
#include <iostream.h>
////////////////////////////mydefs1.h//////////////
class Istack {
      int sz;
      int top;
      int* stack;
public:
      // Using only default constructor
      void init(int ) ;

      void push(int ) ;

      int pop() ;

      bool isempty() ;

      bool isfull() ;
};
//////////////////////////stack1.cc///////////////
// Array implementation of stacks.
// Grow from high index towards 0.
// Using default constructor

void Istack :: init (int nitems = 100)
{
       sz = nitems;
       stack = new int[sz];
       top = sz;
}
void Istack :: push(int x)
{
       assert( !isfull() );
       stack[--top] = x;
}
int Istack :: pop()
{
       assert( !isempty() );
       return( stack[top++] );
}
bool Istack :: isempty()
{
       return( top == sz );
}
bool Istack :: isfull()
{
       return( top == 0 );
}
```

```
//////////////////////stackmain1.cc/////////

int main()
{
      Istack s1;              // default constructor?
            s1.init(500);
      Istack s2;              // default constructor?
            s2.init();        // Perhaps s2.init; too?
      Istack s3();            // a function which returns Istack
      Istack s6;
            s6.init(137);
      Istack s4[5];           // default constructor called 5 times here
      Istack* s5;             // no space, just a pointer

      int i;


      for( i = 0; i < 20; i++ ) {
            s1.push(i);             // s1 is a 500 element stack
      }

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      //          Everything is OK up to here, but there are problems
      //          with both of the following, depending on system used

      s4[3].push(11);               // 3rd stack in the array
      cout << s4[3].pop() << ' ';

      s5 = new Istack;              // get stack space, which constructor?
      s5->push(12);                 // pointing to a stack
      cout << s5->pop() << endl;
      delete s5;                    // explicit removal
      return 0;
}
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Assertion (!isfull()) failed in "stack1.cc" on line 15
*/
//-------------------------------------------
```



```
#include <assert.h>
//////////////////////mydefs1b.cc///////////
class Istack {
      int sz;
      int top;
      int* stack;
public:
      void init(int Nitems = 100)
          { sz = Nitems; stack = new int[sz]; top = sz; }
      void push(int x)
          { assert( !isfull() ); stack[--top] = x; }
      int pop()
          { assert( !isempty() ); return( stack[top++] ); }
      bool isempty()
          { return(( top == sz )); }
      bool isfull()
          { return(( top == 0)); }
};
class Cstack {                      // Changed
      int sz;
      int top;
      char* stack;                  // Changed
public:
                                    // init, push, and pop are changed
      void init(int Nitems = 100)
          { sz = Nitems; stack = new char[sz]; top = sz; }
      void push(char x)
          { assert( !isfull() ); stack[--top] = x; }
      char pop()
          { assert( !isempty() ); return( stack[top++] ); }
      bool isempty()
          { return(( top == sz )); }
      bool isfull()
          { return(( top == 0)); }
};
//////////////////////stackmain1b.cc//////////////
#include <iostream.h>

int main()
{
      Istack s1;                    // a stack of integers
      Cstack s3;                    // a stack of chars
      int i;

      s1.init(500);
      s3.init();                    // default size of 100

      for( i = 0; i < 20; i++ ) {
            s1.push(i);             // also push(s1, i);
      }
```

```
      for( i = 0; i < 20; i++ ) {
           cout << s1.pop() << ' ';
      }
      cout << endl;

      for( i = 0; i < 20; i++ ) {
           s3.push('a' + i);       // overloading push
      }
      for( i = 0; i < 20; i++ ) {
           cout << s3.pop() << ' ';
      }
      cout << endl;
      return 0;
}
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
t s r q p o n m l k j i h g f e d c b a
*/

//--------------------------------------
#include <assert.h>
#include <iostream.h>
////////////////////////mydefs1c.h/////////////
class Istack {
      int sz;
      int top;
      int* stack;
public:
      Istack(void)                  // One creator, no parameters
          { sz = 97; stack = new int[sz]; top = sz;
                cout << "using basic creator 97\n";}
      Istack(int Nitems )           // Another creator
            { sz = Nitems; stack = new int[sz]; top = sz;
              cout << "using pre-set creator " << Nitems << "\n"; }
      ~Istack()                     // The destructor
          { delete[] stack; }
                                    // delete[] removes array space
      void push(int x)
          { assert( !isfull() ); stack[--top] = x; }
      int pop()
          { assert( !isempty() ); return( stack[top++] ); }
      bool isempty()
          { return( top == sz ); }
      bool isfull()
          { return( top == 0); }
};
```



```
////////////////////stackmain1c.cc//////////////

int main()
{
      Istack s1(500);
      Istack s2;              // default constructor?
      Istack s3();            // a function which returns Istack
      Istack s6(137);
      Istack s4[5];           // default constructor called 5 times here
      Istack s9(23);
      Istack* s5;             // no space, just a pointer

//    Istack s8[2](137);      // array of two stacks size 137, NO
//    Istack s7()[2];         // s7() is a function returning an array!
//    Istack s10(23)[2];      // two 23-element stacks?
                              // No way, can only use default
      int i;

      for( i = 0; i < 20; i++ ) {
           s1.push(i);              // s1 is a 500 element stack
      }

      for( i = 0; i < 20; i++ ) {
           cout << s1.pop() << ' ';
      }
      cout << endl;

      s4[3].push(11);               // 3rd stack in the array
      cout << s4[3].pop() << ' ';

      s5 = new Istack;              // get stack space, which constructor?
      s5->push(12);                 // pointing to a stack
      cout << s5->pop() << endl;
      delete s5;                    // explicit removal
      return 0;
}
/*

using pre-set creator 500
using basic creator 97
using pre-set creator 137
using basic creator 97
using basic creator 97
using basic creator 97
using basic creator 97
using basic creator 97
using pre-set creator 23
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
11 using basic creator 97
12
*/
//-----------------------------------
```

```
//////////////////mydefs2b.h///////////////

class node {                        // class node outside Istack, needs friend
      friend class Istack;
      friend ostream& operator << (ostream& , const Istack& ) ;
      int data;
      node* next;
      node(int, node* );            // prototype node constructor
};

class Istack {                      // linked list implementation
      node* top;
      friend ostream& operator << (ostream& , const Istack& ) ;
public:
      void init(int Nitems = 100)
            { top = NULL; }
      void push(int x)
            { top = new node(x, top); }
      int pop();                    // Not defined here, later
      bool isempty()
            { return( top == NULL ); }
      bool isfull()
            { return( FALSE ); }    // Never Full
};

//////////////////////stack2b.cc/////////////
#include <assert.h>
// Linked list implementation of stacks.

node :: node(int d, node* n)
{
      data = d;
      next = n;
}

int Istack :: pop()
{
      assert( !isempty() );
      int temp = top -> data;             // get data
      node* oldtop = top;                 // point to old
      top = top -> next;                  // unlink old
      delete oldtop;                      // discard
      return temp;
}
ostream& operator << (ostream& out, const Istack& right)
{     node* tmp;
      for (tmp = right.top; tmp != NULL; tmp = tmp->next)
            out << tmp->data << ' ';
      out << endl;
      return out;
};
```



```
//////////////////stackmain2b.cc/////////
#include <iostream.h>
int main()
{
      Istack s1, s2;
      int i;

      s1.init(500);
      s2.init();

      for( i = 0; i < 20; i++ ) {
            s1.push(i);
      }

      cout << s1;       // using the new overloaded << operator.

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      s2.push(10);
      i = s2.pop();
      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
*/
//--------------------------------
//////////////////mydefs2a.h/////////////

class Pstack {    // not array of ints but linked list of ints
      class node {
      public:
            int data;
            node* next;
      };
      node* top;
      node* new_node(int , node* );       // prototype only of node initializer

public:
      // Note pop() is a prototype
      void init(int Nitems = 100)
          { top = NULL; }
      void push(int x)
          { top = new_node(x, top); }
      int pop(); // prototype function
      bool isempty()
          { return( top == NULL ); }
      bool isfull()
          { return( FALSE ); }
};
```

```
//////////////////////stackmain2a.cc////////
#include <assert.h>
#include <iostream.h>

// Linked list implementation of stacks.

Pstack :: node* Pstack :: new_node(int d, node* n)
{
      node* t = new node;
      assert( t );
      t -> data = d;
      t -> next = n;
      return( t );
}

int Pstack :: pop()
{
      assert( !isempty() );
      int t = top -> data;
      node* oldtop = top;
      top = top -> next;
      delete oldtop;
      return t;
}
int main()
{
      Pstack s1, s2;
      int i;

      s1.init(500);
      s2.init();

      for( i = 0; i < 20; i++ ) {
            s1.push(i);
      }

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      s2.push(10);
      i = s2.pop();
}
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*/
//-------------------------------------------------
```



```
#include <assert.h>
#include <iostream.h>

class Istack {
      int sz;
      int top;
      int* stack;
public:
      // member function prototypes
      void init(int );
      void push(int );
      int pop();
      bool isempty();
      bool isfull();
};
// Array implementation of stacks.
// Grow from high index towards 0.
void Istack :: init(int nitems = 100)
{
        sz = nitems;
        stack = new int[sz];
        top = sz;
}
void Istack :: push(int x)
{
        assert( !isfull() );
        stack[--top] = x;
}
int Istack :: pop()
{
        assert( !isempty() );
        return( stack[top++] );
}
bool Istack :: isempty()
{
        return( top == sz );
}
bool Istack :: isfull()
{
        return( top == 0 );
}

class TestType {
private:
      int num1, num2;
      float real1;
public:
      TestType(int Num1Start, float Real1Start = 3.14)
                  : num1(Num1Start), real1(Real1Start)
      {
            cout << num1 << " " << num2 << " " << real1 << endl;
      }
};
```

```
class TestNew {
private:
      int x ;

public:
      TestNew()  {}     // Do nothing, but what about x?
};


int main()
{
      Istack s1, s2;
      int i;

//    TestType Object1;                  // no constructor with no params
      TestType Object2(32, 9.45);
      TestType Object3(42);              // one default parameter
//    TestType Object4(52, 7.43, 12);    // no constructor with 3 params
      TestNew ObjectNew;

      int ArraySize = 64;
//    float Array[ArraySize];            // ArraySize must be a const
      float* Array1 = new float[ArraySize];

      cout << "start execution" << endl;
      s1.init(500);
      s2.init();

      for( i = 0; i < 20; i++ ) {
            s1.push(i);
      }

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      s2.push(10);
      i = s2.pop();
}
/*
32 1597 9.45                        // why "1597"?
42 1207960588 3.14                  // why "1207960588"?
start execution
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*/

//-------------------------------------------------
```



```
//////////////////////////stack3.C///////////////
#include <assert.h>
#include <iostream.h>

enum boolean {BAD, GOOD};

class Istack {
      class node {  //linked list stack
      public:
            int data;
            node* next;
      };
      node* top;
      node* new_node(int , node* );

public:
      // Note that Nitems is not used
      void init( int Nitems=100 )
             { top = NULL; }
      void push(int x)
            {top = new_node(x, top); }
      int pop();
      boolean isempty()
            { return ((top == NULL) ? BAD : GOOD); }
      boolean isfull()
            { return (BAD); }
};


// Linked list implementation of stacks.

Istack :: node* Istack :: new_node(int d, node* n)
{
      node* t = new node;
      assert( t );            // space found OK
      t -> data = d;
      t -> next = n;
      return( t );            // return pointer to stack
}

int Istack :: pop()
{
      assert( isempty() == GOOD);
      int t = top -> data;
      node* oldtop = top;
      top = top -> next;
      delete oldtop;          // removes node that held
      return t;               // data item being returned
}
```

```
int main()
{
      Istack s1,  s2;
      int i;

      s1.init(500);
      s2.init();

      for( i = 0; i < 20; i++ ) {
            s1.push(i);
      }

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      s2.push(10);
      i = s2.pop();
}
//-------------------------------------------------
#include <assert.h>
#include <iostream.h>
//////////////////////mydefs2.h////////////////
class node {

      friend class Istack;

      int data;
      node* next;
      node(int d, node* n)
      {
            data = d;
            next = n;
      }     // constructor and initializer of node
};
```



```
class Istack {                                  // linked list implementation
      node* top;
public:
      void init(int Nitems = 100)
          { top = NULL; }
      void push(int x)
          { top = new node(x, top); }
      int  pop()
      {
            assert( !isempty() );
            int temp = top -> data;             // get data
            node* oldtop = top;                 // point to old
            top = top -> next;                  // unlink old
            delete oldtop;                      // discard
            return temp;
      }
      bool isempty()
          { return( top == NULL ); }
      bool isfull()
          { return( FALSE ); }                  // Never Full
};

//////////////////////stackmain2.cc////////

int main()
{
      Istack s1, s2;
      int i;

      s1.init(500);
      s2.init();

      for( i = 0; i < 20; i++ ) {
            s1.push(i);
      }

      for( i = 0; i < 20; i++ ) {
            cout << s1.pop() << ' ';
      }
      cout << endl;

      s2.push(10);
      i = s2.pop();
}
/*
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
*/
//----------------------------------
```