```cpp
// stack.cpp -- Stack member functions
#include "stack.h"
Stack::Stack()    // create an empty stack
{
    top = 0;
}

bool Stack::isempty() const
{
    return top == 0;
}

bool Stack::isfull() const
{
    return top == MAX;
}

bool Stack::push(const Item & item)
{
    if (top < MAX)
    {
        items[top++] = item;
        return true;
    }
    else
        return false;
}

bool Stack::pop(Item & item)
{
    if (top > 0)
    {
        item = items[--top];
        return true;
    }
    else
        return false;
}
```

```cpp
// stacker.cpp -- testing the Stack class
#include <iostream>
#include <cctype>  // or ctype.h
#include "stack.h"
int main()
{
    using namespace std;
    Stack st; // create an empty stack
    char ch;
    unsigned long po;
    cout << "Please enter A to add a purchase order,\n"
        << "P to process a PO, or Q to quit.\n";
    while (cin >> ch && toupper(ch) != 'Q')
    {
        while (cin.get() != '\n')
            continue;
        if (!isalpha(ch))
```

```
        {
            cout << '\a';
            continue;
        }
        switch(ch)
        {
             case 'A':
             case 'a': cout << "Enter a PO number to add: ";
                       cin >> po;
                       if (st.isfull())
                           cout << "stack already full\n";
                       else
                           st.push(po);
                       break;
             case 'P':
             case 'p': if (st.isempty())
                           cout << "stack already empty\n";
                       else {
                           st.pop(po);
                           cout << "PO #" << po << " popped\n";
                       }
                       break;
        }
        cout << "Please enter A to add a purchase order,\n"
             << "P to process a PO, or Q to quit.\n";
    }
    cout << "Bye\n";
    return 0;
}




// stock1.cpp SPA Stock class implementation with constructors, destructor added
#include <iostream>
#include "stock1.h"

// constructors (verbose versions)
Stock::Stock()        // default constructor
{
    std::cout << "Default constructor called\n";
    std::strcpy(company, "no name");
    shares = 0;
    share_val = 0.0;
    total_val = 0.0;
}

Stock::Stock(const char * co, int n, double pr)
{
    std::cout << "Constructor using " << co << " called\n";
    std::strncpy(company, co, 29);
    company[29] = '\0';

    if (n < 0)
    {
        std::cerr << "Number of shares can't be negative; "
                  << company << " shares set to 0.\n";
        shares = 0;
    }
    else
```

```
        shares = n;
    share_val = pr;
    set_tot();
}
// class destructor
Stock::~Stock()        // verbose class destructor
{
    std::cout << "Bye, " << company << "!\n";
}

// other methods
void Stock::buy(int num, double price)
{
     if (num < 0)
    {
        std::cerr << "Number of shares purchased can't be negative. "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares += num;
        share_val = price;
        set_tot();
    }
}

void Stock::sell(int num, double price)
{
    using std::cerr;
    if (num < 0)
    {
        cerr << "Number of shares sold can't be negative. "
             << "Transaction is aborted.\n";
    }
    else if (num > shares)
    {
        cerr << "You can't sell more than you have! "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares -= num;
        share_val = price;
        set_tot();
    }
}

void Stock::update(double price)
{
    share_val = price;
    set_tot();
}

void Stock::show()
{
    using std::cout;
    using std::endl;
    cout << "Company: " << company
        << "  Shares: " << shares << endl
        << "  Share Price: $" << share_val
```

```
        << "  Total Worth: $" << total_val << endl;
}



// stock2.cpp -- improved version
#include <iostream>
#include "stock2.h"

// constructors
Stock::Stock()        // default constructor
{
    std::strcpy(company, "no name");
    shares = 0;
    share_val = 0.0;
    total_val = 0.0;
}

Stock::Stock(const char * co, int n, double pr)
{
    std::strncpy(company, co, 29);
    company[29] = '\0';

    if (n < 0)
    {
        std::cerr << "Number of shares can't be negative; "
                  << company << " shares set to 0.\n";
        shares = 0;
    }
    else
        shares = n;
    share_val = pr;
    set_tot();
}

// class destructor
Stock::~Stock()        // quiet class destructor
{
}

// other methods
void Stock::buy(int num, double price)
{
     if (num < 0)
    {
        std::cerr << "Number of shares purchased can't be negative. "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares += num;
        share_val = price;
        set_tot();
    }
}

void Stock::sell(int num, double price)
{
    using std::cerr;
```

```
    if (num < 0)
    {
        cerr << "Number of shares sold can't be negative. "
             << "Transaction is aborted.\n";
    }
    else if (num > shares)
    {
        cerr << "You can't sell more than you have! "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares -= num;
        share_val = price;
        set_tot();
    }
}

void Stock::update(double price)
{
    share_val = price;
    set_tot();
}

void Stock::show() const
{
    using std::cout;
    using std::endl;
    cout << "Company: " << company
        << "  Shares: " << shares << endl
        << "  Share Price: $" << share_val
        << "  Total Worth: $" << total_val << endl;
}

const Stock & Stock::topval(const Stock & s) const
{
    if (s.total_val > total_val)
        return s;
    else
        return *this;
}



// usestok1.cpp -- use the Stock class
#include <iostream>
#include "stock1.h"

int main()
{
    using std::cout;
    using std::ios_base;
    cout.precision(2);                              // #.## format
    cout.setf(ios_base::fixed, ios_base::floatfield);// #.## format
    cout.setf(ios_base::showpoint);                 // #.## format

    cout << "Using constructors to create new objects\n";
    Stock stock1("NanoSmart", 12, 20.0);            // syntax 1
    stock1.show();
    Stock stock2 = Stock ("Boffo Objects", 2, 2.0); // syntax 2
```

```
    stock2.show();

    cout << "Assigning stock1 to stock2:\n";
    stock2 = stock1;
    cout << "Listing stock1 and stock2:\n";
    stock1.show();
    stock2.show();

    cout << "Using a constructor to reset an object\n";
    stock1 = Stock("Nifty Foods", 10, 50.0);    // temp object
    cout << "Revised stock1:\n";
    stock1.show();
    cout << "Done\n";
    return 0;
}




// usestok2.cpp -- use the Stock class
// compile with stock2.cpp
#include <iostream>
#include "stock2.h"

const int STKS = 4;
int main()
{
    using std::cout;
    using std::ios_base;

// create an array of initialized objects
    Stock stocks[STKS] = {
        Stock("NanoSmart", 12, 20.0),
        Stock("Boffo Objects", 200, 2.0),
        Stock("Monolithic Obelisks", 130, 3.25),
        Stock("Fleep Enterprises", 60, 6.5)
        };

    cout.precision(2);                                  // #.## format
    cout.setf(ios_base::fixed, ios_base::floatfield);// #.## format
    cout.setf(ios_base::showpoint);                     // #.## format

    cout << "Stock holdings:\n";
    int st;
    for (st = 0; st < STKS; st++)
        stocks[st].show();

    Stock top = stocks[0];
    for (st = 1; st < STKS; st++)
        top = top.topval(stocks[st]);
    cout << "\nMost valuable holding:\n";
    top.show();

    return 0;
}
```

```
361
362
363  // stocks.cpp -- the whole program
364  #include <iostream>
365  #include <cstring>
366
367  class Stock  // class declaration
368  {
369  private:
370      char company[30];
371      int shares;
372      double share_val;
373      double total_val;
374      void set_tot() { total_val = shares * share_val; }
375  public:
376      void acquire(const char * co, int n, double pr);
377      void buy(int num, double price);
378      void sell(int num, double price);
379      void update(double price);
380      void show();
381  };    // note semicolon at the end
382
383  void Stock::acquire(const char * co, int n, double pr)
384  {
385      std::strncpy(company, co, 29);  // truncate co to fit company
386      company[29] = '\0';
387      if (n < 0)
388      {
389          std::cerr << "Number of shares can't be negative; "
390                    << company << " shares set to 0.\n";
391          shares = 0;
392      }
393      else
394          shares = n;
395      share_val = pr;
396      set_tot();
397  }
398
399  void Stock::buy(int num, double price)
400  {
401       if (num < 0)
402      {
403          std::cerr << "Number of shares purchased can't be negative. "
404               << "Transaction is aborted.\n";
405      }
406      else
407      {
408          shares += num;
409          share_val = price;
410          set_tot();
411      }
412  }
413
414  void Stock::sell(int num, double price)
415  {
416      using std::cerr;
417      if (num < 0)
418      {
419          cerr << "Number of shares sold can't be negative. "
420               << "Transaction is aborted.\n";
```

```
    }
    else if (num > shares)
    {
        cerr << "You can't sell more than you have! "
             << "Transaction is aborted.\n";
    }
    else
    {
        shares -= num;
        share_val = price;
        set_tot();
    }
}

void Stock::update(double price)
{
    share_val = price;
    set_tot();
}

void Stock::show()
{
    using std::cout;
    using std::endl;
    cout << "Company: " << company
        << "  Shares: " << shares << endl
        << "  Share Price: $" << share_val
        << "  Total Worth: $" << total_val << endl;
}

int main()
{
    using std::cout;
    using std::ios_base;

    Stock stock1;
    stock1.acquire("NanoSmart", 20, 12.50);
    cout.setf(ios_base::fixed);        // #.## format
    cout.precision(2);                 // #.## format
    cout.setf(ios_base::showpoint);    // #.## format
    stock1.show();
    stock1.buy(15, 18.25);
    stock1.show();
    stock1.sell(400, 20.00);
    stock1.show();
    return 0;
}
```