3 Exercises

1. Rewrite the Stock class introduced in lecture10,program example **stock02.h**, to use dynamically allocated memory directly instead of using string class objects to hold the stock names. Also replace the show() member function with an overloaded operator<<() definition. Write a program to test the new definition.

```
// stock02.h -- augmented version
#ifndef STOCK20_H_
#define STOCK20_H_
#include <string>
class Stock
{
    private:
    std::string company;
    int shares;
    double share_val;
    double total_val;
    void set_tot() { total_val = shares * share_val; }

    public:
    Stock(); // default constructor
    Stock(const std::string & co, long n = 0, double pr = 0.0);
    ~Stock(); // do-nothing destructor

    void buy(long num, double price);
    void sell(long num, double price);
    void update(double price);
    void show()const;
    const Stock & topval(const Stock & s) const;
};
#endif
```

2. The declaration of Stack as follows:

```
// stack.h -- class declaration for the stack ADT
typedef unsigned long Item;

class Stack
{

private:
    enum {MAX = 10};      // constant specific to class
    Item  * pitems;       // holds stack items
    int size;             // number of elements in stack
    int top;              // index for top stack item
public:
    Stack(int n = MAX);    // creates stack with n elements
    Stack(const Stack & st);
    ~Stack();
    bool isempty() const;
    bool isfull() const;
    // push() returns false if stack already is full, true otherwise
    bool push(const Item & item); // add item to stack
    // pop() returns false if stack already is empty, true otherwise
    bool pop(Item & item);  // pop top into item
    Stack & operator=(const Stack & st);
};
```

Implement all the methods and write a program to demonstrates all the methods, including copy constructor and assignment operator.