```
Script started on Mon 05 Nov 2012 10:23:15 AM CST
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ p
wd
/home/georgia/cplusplus
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP --version
This is CPP version 1.219 executing under perl v5.12.4 and compiling with:

g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at dou\033[Kblelist.info
Name: Jakob Hansen
Class: CSC122 - Evening Section
Lab: how many items in your list?

Levels attempted:

3.5 - For just getting the basic instructions down.

+ 1.5 for making list growth automatic: whenever the list exceeds it’s allocated
space, it allocates twice as much space, growing exponentially.

+ 2.5 for overloading operators: <<, >>, and two versions of []: accessor and
mutator.

Program Description:

A class which implements a dynamic array of doubles. List automatically grows
to accomodate however many doubles need to be added at that moment, and supplies
some nice overloaded operators to make life easier!
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at dublist.h
#ifndef DUBLIST_H_INC
#define DUBLIST_H_INC

#include <iostream>
#include <cstddef>

using namespace std;

class doublelist
{
    double * p;  // array of doubles
    size_t current_size;    // number of array positions filled
    size_t capacity;     // number of POSSIBLE array positions

    bool reallocate(size_t more);

public:


    doublelist(void) : p(NULL), current_size(0), capacity(0) { }

    doublelist(size_t caller_size) : p(new double[caller_size]),
                  current_size(0), capacity(caller_size)
    {
        capacity = p == NULL ? 0 : caller_size;
    }

    doublelist(const doublelist & dl);

    ˜doublelist(void) { delete [] p; p = NULL; }

    doublelist & operator=(const doublelist & dl);

    double operator[](const size_t index) const
    {
        return (p == NULL || index >= current_size) ? 0.0 : p[index];
    }

    double & operator[](size_t index)
    {
        double num;
        num = 0.0;
        return (p == NULL || index >= current_size) ? num : p[index];
    }


    void insert_double(double num);

    double get_last(void) const { return p == NULL ? 0.0 : p[current_size-1]; }

    void delete_last(void)
    {
        if (p != NULL)
        {
            current_size--;
        }
        return;
    }

    size_t get_size(void) const { return p == NULL ? 0 : current_size; }

    void print(void) const;

};

istream & operator>>(istream & in, doublelist & dl);

ostream & operator<<(ostream & out, const doublelist & dl);




#endif
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at dublist.cpp
#include "dublist.h"
#include <cstddef>
#include <iostream>
#include <climits>

using namespace std;


doublelist::doublelist(const doublelist & dl)
          : p(new double[dl.capacity]),
            current_size(dl.current_size),
            capacity(dl.capacity)
{
    if (p != NULL)
    {
        for (size_t i=0; i != current_size; i++)
        {
            p[i] = dl.p[i];
        }
    }
    else
    {
        current_size = capacity = 0;
    }
}
```

```
doublelist & doublelist::operator=(const doublelist & dl)
{
    if ( &dl != this )
    {
        delete [] p;

        p = new double[dl.capacity];

        if (p != NULL)
        {
            capacity = dl.capacity;
            current_size = dl.current_size;
            for(size_t i=0; i != current_size; i++)
            {
                p[i] = dl.p[i];
            }
        }
        else
        {
            current_size = capacity = 0;
        }
    }
    return *this;
}

bool doublelist::reallocate(size_t more)
{
    double * pnew;
    bool okay = false;
    pnew = new double [capacity + more];
    if (pnew != NULL)
    {
        okay = true;
        for (size_t i=0; i != current_size; i++)
        {
            pnew[i] = p[i];
        }
        delete [] p;
        p = pnew;
        pnew = NULL;
        capacity += more;
    }
    return okay;
}


void doublelist::insert_double(double num)
{
    if (current_size == capacity)
    {
        if (reallocate(p == NULL ? 2 : capacity))
        {
            p[current_size++] = num;
        }
        else
        {
            cerr << "\nERROR: MEMORY ALLOCATION FAILED";
        }
    }
    else
    {
        p[current_size++] = num;
    }
    return;
}


void doublelist::print(void) const
{
    size_t i;
    for (i = 0; i < current_size; i++)
    {
        cout << p[i] << ' ';
    }
    return;
}

istream & operator>>(istream & in, doublelist & dl)
{
    double num;
    in >> num;
    while (in.peek() != '\n' && !in.fail())
    {
        dl.insert_double(num);
        in >> num;
    }
    in.clear();
    in.ignore(INT_MAX, '\n');

    return in;
}


ostream & operator<<(ostream & out, const doublelist & dl)
{
    size_t i;
    for (i=0; i != dl.get_size(); i++)
    {
        out << dl[i] << ' ';
    }
    return out;
}

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at doubletest.\033[Kcpp
#include "dublist.h"
#include <iostream>
#include <cstddef>
#include <climits>

using namespace std;

int main(void)
{

    doublelist dlist, dlist2;

    cout << "\nDynamic list of Doubles!\n";

    /*cout << "\nWhat size do you want your list to be? ";

    cin >> size;

    cout << "\nOk, a list of size: " << size;

    dlist = doublelist(size);*/

    cout << "\nEnter a bunch of doubles! ";

    cin >> dlist;

    cout << "\nThe Current number of filled spots in dlist is: "
         << dlist.get_size();

    cout << "\nThe Last number in dlist is: " << dlist.get_last();

    cout << "\nDeleting last number in dlist - the size is: ";
    dlist.delete_last();

    cout << dlist.get_size()
         << "\nThe last number in dlist is now: " << dlist.get_last();

    cout << "\nThe whole dlist is: " << dlist;


    cout << "\n[] operator test: calling dlist[6]:\n";

    cout << "\ndlist[6] is: " << dlist[6] << '\n';

    cout << "\nTrying dlist[6] = 3.5";

    dlist[6] = 3.5;

    cout << "\ndlist[6] is now: " << dlist[6];

    cout << "\nEnter a bunch of doubles! ";

    cin >> dlist2;

    cout << "\ndlist2 is: " << dlist2 << '\n';


    cout << "\nCopying dlist into dlist2...";

    dlist2 = dlist;

    cout << "\ndlist2 is now: " << dlist2 << '\n';


    cout << "\n\nBye!";
    return 0;
}
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP doubletest \033[Kdublist
doubletest.cpp***
dublist.cpp...


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/doubletest.out

Dynamic list of Doubles!

Enter a bunch of doubles! 1.22 2.3 3.4 4.5 5.6 6.7 7.8 8.9 9.10 10.11 11.12 q

The Current number of filled spots in dlist is: 11
The Last number in dlist is: 11.12
Deleting last number in dlist - the size is: 10
The last number in dlist is now: 10.11
The whole dlist is: 1.22 2.3 3.4 4.5 5.6 6.7 7.8 8.9 9.1 10.11
[] operator test: calling dlist[6]:

dlist[6] is: 7.8

Trying dlist[6] = 3.5
dlist[6] is now: 3.5
Enter a bunch of doubles! 12.11 11.10 10.9 9.8 8.7 7.6 6.5 5.4 4.3 3.2 q

dlist2 is: 12.11 11.1 10.9 9.8 8.7 7.6 6.5 5.4 4.3 3.2

Copying dlist into dlist2...
dlist2 is now: 1.22 2.3 3.4 4.5 5.6 6.7 3.5 8.9 9.1 10.11


Bye!\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplu
s$ exit
exit

Script done on Mon 05 Nov 2012 10:25:03 AM CST
```