Byron maungu

lab 3

vector of integers

cse 455

```
#include <iostream>
#include  "svector.h"

using namespace std;
int main()
{
   vector <int> example;
   example.push_back(3);        //add 3 onto the vector
   example.push_back(10);       //add 10 to the end
   example.push_back(33);       //add 33 to the end
   for(int x=0; x<example.size(); x++)
   {
      cout<<example[x]<<" ";    //output: 3 10 33
   }
   if(!example.empty())
      example.clear();
   vector <int> another_vector;
   another_vector.push_back(10); //adds to end of vector
   example.push_back(10);
   if(example==another_vector)
   {
      example.push_back(20);
   }
   for(int y=0; y<example.size(); y++)
   {
```

```
    cout<<example[y]<<" ";    //should output 10 20
}
return 0;
```

Test unit

```
ifndef sVector_h
#define sVector_h



template <class T> class vector {
public:
        typedef T * iterator;

                // constructors
        vector   () { buffer = 0; reserve(0); }
        vector   (unsigned int size) { buffer = 0; resize(size); }
        vector   (unsigned int size, T initial)

                                { buffer = 0; resize(size); fill(begin(), end(), initial); }
        vector   (vector & v)
                                { buffer = 0; resize(v.size()); copy(v.begin(), v.end(), begin());}
        ~vector ()          { delete buffer; }

                // member functions
        T                       back     ()         { return buffer[mySize-1]; }
        iterator begin    ()         { return buffer; }
        int                     capacity          () { return myCapacity; }sVector_h
        iterator end              ()         { return buffer + mySize; }
        bool              empty   ()         { return mySize == 0; }
        T                       front    ()         { return buffer[0]; }
        void              pop_back () { mySize--; }
        void              push_back        (T value);
```

```
        void            reserve (unsigned int newSize);
        void            resize  (unsigned int newSize)  { reserve(newSize); mySize = newSize; }
        int                     size    ()      { return mySize; }
                // operators
        T &             operator [ ] (unsigned int index) { return buffer[index]; }
private:
        unsigned int mySize;
        unsigned int myCapacity;
        T * buffer;
};

template <class T> void vector<T>::reserve (unsigned int newCapacity)
{
        if (buffer == 0) {
                mySize = 0;
                myCapacity = 0;
                }
        if (newCapacity <= myCapacity)
                return;
        T * newBuffer = new T [newCapacity];
        copy (buffer, buffer + mySize, newBuffer);
        myCapacity = newCapacity;
        delete buffer;
        buffer = newBuffer;
}

template <class T> void vector<T>::push_back (T value)
{
        if (mySize >= myCapacity)
```

```
                reserve(myCapacity + 5);
        buffer[mySize++] = value;
}


# endif
```