#include <iostream>

**using** **namespace** std;

*//Template to allow multiple types of elements in following class*

**template**<**class** element>

**class** DynamicStringArray

{

**private**:

element \*dynamicArray;

**int** size;

**public**:

DynamicStringArray();

DynamicStringArray(**int** s);

**int** getSize();

**void** addEntry(element entry);

**bool** deleteEntry(element entry);

element getEntry(**const** **int** n);

DynamicStringArray(**const** DynamicStringArray<element>& copy);

DynamicStringArray<element>& **operator**==(**const** DynamicStringArray<element>& other);

~DynamicStringArray();

};

*//(Default constructor*

**template**<**class** element>

DynamicStringArray<element>::DynamicStringArray()

{

dynamicArray=**NULL**;

size=0;

}

*//Constructor taking parameter*

**template**<**class** element>

DynamicStringArray<element>::DynamicStringArray(**int** s)

{

dynamicArray= **new** element[s];

size=s;

}

*//Accessor function to return size*

**template**<**class** element>

**int** DynamicStringArray<element>::getSize()

{

**return** size;

}

*//Function to add element to end of array*

**template**<**class** element>

**void** DynamicStringArray<element>::addEntry(element entry)

{

element \*newDynamicArray= **new** element[size+1];

**int** i;

**for**(i=0;i<size;i++)

{

newDynamicArray[i]=dynamicArray[i];

}

newDynamicArray[i]=entry;

size++;

**delete**[] dynamicArray;

dynamicArray=newDynamicArray;

}

*//Function to delete element from array*

**template**<**class** element>

**bool** DynamicStringArray<element>::deleteEntry(element entry)

{

**int** i;

**for**(i=0; i<size; i++)

**if**(dynamicArray[i].compare(entry)==0)

**break**;

**if**(i==size)

**return** **false**;

element \*newDynamicArray= **new** element[size-1];

**int** j=0;

**for**(i=0; i<size; i++)

{

**if**(dynamicArray[i].compare(entry)!=0)

newDynamicArray[j++]=dynamicArray[i];

}

**delete**[] dynamicArray;

size--;

dynamicArray= newDynamicArray;

**return** **true**;

}

*//Function that returns element of array at specified position*

**template**<**class** element>

element DynamicStringArray<element>::getEntry(**int** n)

{

**if**(n<0 || n>size)

**return** **NULL**;

**return** dynamicArray[n];

}

*//Copy constructor*

**template**<**class** element>

DynamicStringArray<element>::DynamicStringArray(**const** DynamicStringArray<element>& copy)

{

size=copy.size;

dynamicArray= **new** element[size];

**for**(**int** i=0; i<size; i++)

dynamicArray[i]= copy.dynamicArray[i];

}

*//Assignment operator*

**template**<**class** element>

DynamicStringArray<element>& DynamicStringArray<element>::**operator**==(**const** DynamicStringArray<element>& other)

{

**if** (dynamicArray != **NULL**)

{

**delete**[] dynamicArray;

}

**if** (other.size == 0)

{

size = 0;

dynamicArray = **NULL**;

}

**else**

{

size = other.size;

dynamicArray = **new** element[size];

**for** (**int** i = 0; i < size; i++)

{

dynamicArray[i] = other.dynamicArray[i];

}

}

**return** (\***this**);

}

*//Destructor*

**template**<**class** element>

DynamicStringArray<element>::~DynamicStringArray()

{

**delete**[] dynamicArray;

}

**int** main()

{

DynamicStringArray<string> names;

*// List of names*

names.addEntry("Frank");

names.addEntry("Wiggum");

names.addEntry("Nahasapeemapetilon");

names.addEntry("Quimby");

names.addEntry("Flanders");

*// Output list*

cout << "List of names:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

*// Add and remove some names*

names.addEntry("Spuckler");

cout << "After adding a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.deleteEntry("Nahasapeemapetilon");

cout << "After removing a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.deleteEntry("Skinner");

cout << "After removing a name that isn't on the list:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.addEntry("Muntz");

cout << "After adding another name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

*// Remove all of the names by repeatedly deleting the last one*

**while** (names.getSize() > 0) {

names.deleteEntry(names.getEntry(names.getSize() - 1));

}

cout << "After removing all of the names:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

names.addEntry("Olivia");

cout << "After adding a name:" << endl;

**for** (**int** i = 0; i < names.getSize(); i++)

cout << names.getEntry(i) << endl;

cout << endl;

cout << "Testing copy constructor" << endl;

DynamicStringArray<string> names2(names);

*// Remove Olivia from names*

names.deleteEntry("Olivia");

cout << "Copied names:" << endl;

**for** (**int** i = 0; i < names2.getSize(); i++)

cout << names2.getEntry(i) << endl;

cout << endl;

cout << "Testing assignment" << endl;

DynamicStringArray<string> names3;

names3=names2;

*// Remove Olivia from names2*

names2.deleteEntry("Olivia");

cout << "Copied names:" << endl;

**for** (**int** i = 0; i < names3.getSize(); i++)

cout << names3.getEntry(i) << endl;

cout << endl;

cout << "Testing dynamic array of ints" << endl;

DynamicStringArray<**int**> nums;

nums.addEntry(10);

nums.addEntry(20);

nums.addEntry(30);

**for** (**int** i = 0; i < nums.getSize(); i++)

cout << nums.getEntry(i) << endl;

cout << endl;

cout << "Enter a character to exit." << endl;

**char** wait;

cin >> wait;

**return** 0;

}

