```
#include<iostream>
#include<string>
#include<cstdlib>

using namespace std;

/**
Most of these comments were taken from the previous assignment,
The only new comments are those that explain the implementation of structures.
**/

//initial template initialization for the entire class
template <class item>
class DynamicArray //string dynamicarray has been changed to class dynamic array to allow for general usage
{

public:
  DynamicArray()
  {
    //constructor that sets the pointer to null and sizer to 0
    dynamicArray = nullptr;
    size = 0;
  }

  DynamicArray(const DynamicArray& dsa)
  {
    //Creates a new array with size +1
    size = dsa.size;
    dynamicArray = new item[size]; //creates a new dynamic array with an item property
    for(int i = 0; i<size; i++)
    {
      //reassign values
      dynamicArray[i] = dsa.dynamicArray[i];
    }
  }

  int getSize()
  {
    //returns the size of the array
    return size;
  }

  void addEntry(item user)
  {
```

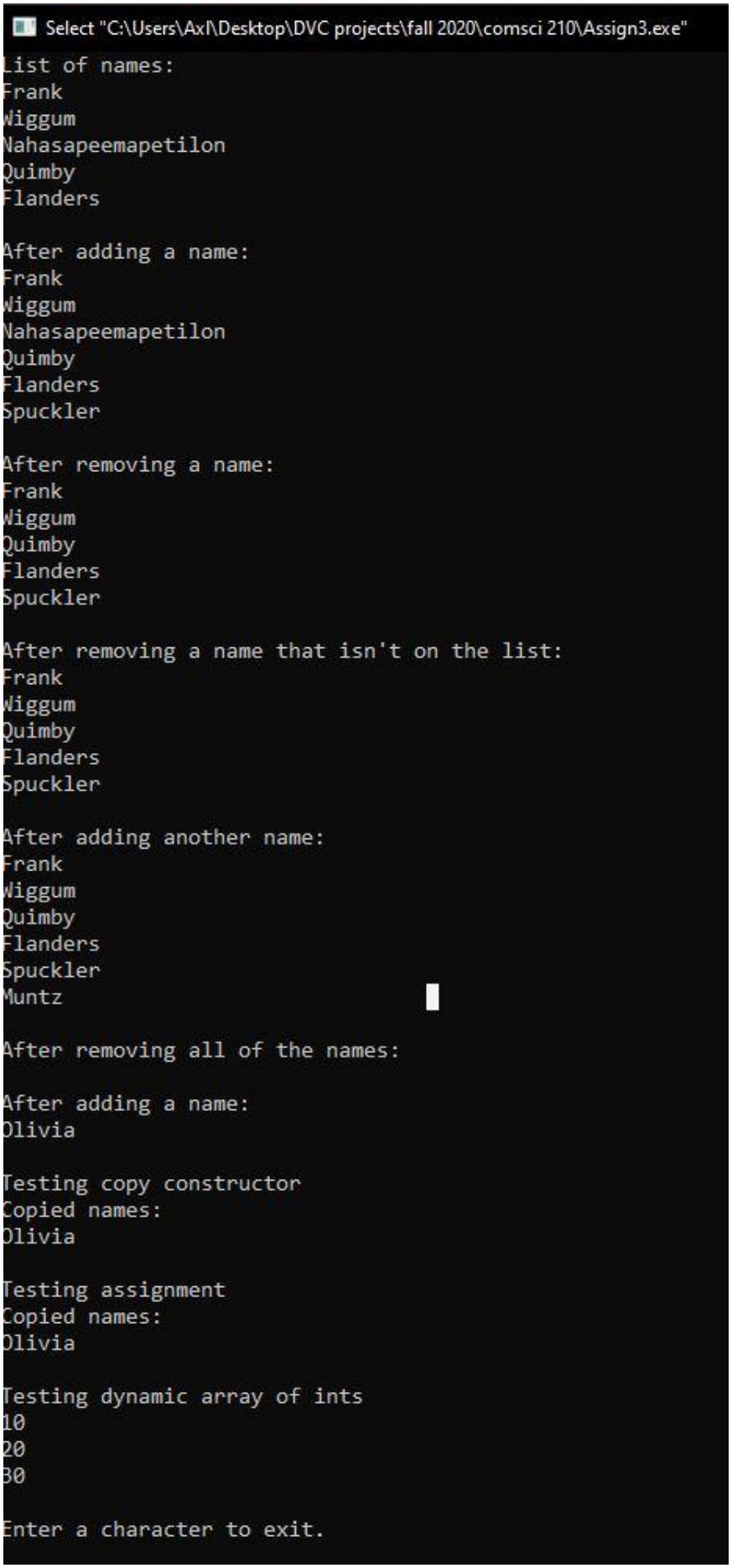
```
    //recreates a new dynamic array with an increased size
    item* newArray = new item[size+1]; //creates a new 'item' instead of the previous string
    size = size + 1;
    int i;
    for (i=0; i<size-1; i++)
    {
        //loop to assign the values to the new array
        newArray[i] = dynamicArray[i];
    }
    newArray[size-1] = user;
    delete[] dynamicArray;
    dynamicArray = newArray;
}

bool deleteEntry(item input)
{
    int j; //counter for the loop
    for(j = 0; j<size; j++)
    {
        if(dynamicArray[j] == input)
        {
            break;
            //breaks the loop when the input is found
        }
    }

    if(j==size)
    {
        //otherwise return false from the function
        return false;
    }

    //create a new dynamic array with a size that is 1 less
    item* newArray = new item[size-1];

    int l = 0;

    for(int k=0; k<size; k++)
    {
        if(dynamicArray[k]!=input)
        {
            //assigns every value that is not equal to the user input
            newArray[l++] = dynamicArray[k];
        }
    }
```

```
    //delete the dynamic array
    delete[] dynamicArray;
    //reduce the size
    size--;
    //reassign the new array to the dynamic array
    dynamicArray = newArray;
    //return true after recreating the arrays
    return true;

  }

  item getEntry(int input)
  {
    //get entry and return the value when the input is considered valid
    if(input<size && input>=0)
    {
      return dynamicArray[input];
    }
    else
    {
      return NULL; //if invalid input, return nothing
    }
  }

  DynamicArray operator==(const DynamicArray& dsa)
  {
    //operator overloading for the == assignment operator
    size = dsa.size;
    dynamicArray = new item[size]; //now an "item" instead of string
    for(int i=0; i<size; i++)
    {
      dynamicArray[i] = dsa.dynamicArray[i];
    }
    return *this;
    //return the submitted input

  }

  ~DynamicArray()
  {
    //deconstructor that deletes the array
    delete[] dynamicArray;
  }

private:
```

```
   //private declarations for certain variables
   item *dynamicArray; //the variable type has been changed to item to accomodate for the
template
   int size;

};




//test driver function required form the assignment
int main()
{
DynamicArray<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;
DynamicArray<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;
DynamicArray<string> 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;
DynamicArray<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;
}
```


Select "C:\Users\Axl\Desktop\DVC projects\fall 2020\comsci 210\Assign3.exe"
List of names:
Frank
Wiggum
Nahasapeemapetilon
Quimby
Flanders
After adding a name:
Frank
Wiggum
Nahasapeemapetilon
Quimby
Flanders
Spuckler
After removing a name:
Frank
Wiggum
Quimby
Flanders
Spuckler
After removing a name that isn't on the list:
Frank
Wiggum
Quimby
Flanders
Spuckler
After adding another name:
Frank
Wiggum
Quimby
Flanders
Spuckler
Muntz
After removing all of the names:
After adding a name:
Olivia
Testing copy constructor
Copied names:
Olivia
Testing assignment
Copied names:
Olivia
Testing dynamic array of ints
10
20
30
Enter a character to exit.