In this application, we will verify that when we run the program `SimpleVectrTest.cpp`, there are no memory leaks. Every time an object is created or destroyed, we will print a message giving the address of the object. This will show that every object that is created by a constructor is later destroyed by the destructor.

In the file `vectr.cpp`, in the constructors with prototypes

```cpp
    Vectr::Vectr(int);
    Vectr::Vectr(int, double);
```

rewrite the statements

```cpp
    else
        element = new double[size];
```

as

```cpp
    else
    {
        element = new double[size];
        cout << "Constructed vector at " << this << endl;
    }
```

Run `SimpleVectrTest.cpp`. Copy the output and paste it into a Word document named `Lab 4 Output`. The output should identify every instance in which a `Vectr` is either created or destroyed. Match them up according to the addresses of the vectors. Do it in the following manner:

1. Starting at the beginning of the output, at the beginning of those lines in which objects are constructed, label the lines A, B, C, . . ., sequentially throughout the output.

2. Starting at the beginning of the output, at the beginning of those lines in which objects are destroyed, match the destroyed object with the line in which that object was constructed by using the labels used earlier (A, B, C, . . .).

3. Verify that every vector that was created was later destroyed.

Now find every place where memory is allocated or deallocated dynamically and print a message giving the address of the memory. In other words, find every place where `new` or `delete` is called. Print the message

```cpp
    cout << "Allocated memory at " << element << endl;
```

after every usage of the `new` operator and

```cpp
    cout << "Deallocated memory at " << element << endl;
```
before each usage of the `delete` operator. Make similar changes in the functions `input()` (where `new` is used) and `setSize()` (where both `new` and `delete` are used). Print the address of the deleted memory only if the pointer is not null. When memory is deallocated, be sure to print the address before using `delete` and when memory is allocated, but sure to print the address after using `new`.

Then by matching the deallocations with the allocations, we will see that no memory was leaked.

1. Starting at the beginning of the output, at the beginning of those lines in which dynamic memory is allocated, label the lines 1, 2, 3, . . . , sequentially throughout the output.

2. Starting at the beginning of the output, at the beginning of those lines in which dynamic memory is deallocated, match the deallocated memory with the line in which that memory was allocated by using the labels used earlier (1, 2, 3, . . .).

3. Verify that all allocated memory was deallocated and that every created object was destroyed.