
1 Class Description

Design and implement an extensible vector class named `ExtVector`. The extensible vector class is similar to the `Vector` class of lab 9; however, the extensible vector class must dynamically change its size to store a varying number of elements (a dynamic array, where each element is a double). The class should have the appropriate constructors, destructor, and other methods for managing vector coordinates. Since `ExtVector` has a pointer member, be certain **no** memory leaks occur. The `ExtVector` class must have the following member functions.

1.1 Constructors

Implement (at least) the following constructors. Other constructors may be necessary.

- Null constructor - Empty vector
- Copy constructor - Makes a deep copy of an `ExtVector`
- Vector constructor - Create an `ExtVector` from a `Vector`.

1.2 Destructor

Properly delete the extensible vector.

1.3 Size of the Vector

The member function `size()` must return the number of elements in the vector.

1.4 Adding and Deleting Elements

Implement the following two methods to add or remove elements from the vector.

- `append(double value)` - Adds a new element with value `value` to the end of the vector. For example, the following code segment would create a vector with the values `{4.0, 5.5, 7.2}`.

```
ExtVector a; // null constructor
a.append(4.0);
a.append(5.5);
a.append(7.2);
```

- `removeItem(int n)` - Removes the n^{th} element from the vector (assume the vector index starts with one, not zero). If `n` is out-of-bounds print an error message to the user.

1.5 Assignment Operator

Make a deep copy of the RHS `ExtVector` (or `Vector`) and assign it to the LHS `ExtVector`.

1.6 +/- Operator

Addition and subtraction of the `ExtVector` can occur with the following data types. Note the behavior of the operation depends on the data type. All operations must be **symmetric**. Always return an `ExtVector`.

- `double` - As with the `Vector` class, the scalar value should be added/subtracted to each element. For example if $\vec{a} = \{1.0, 2.0, 4.0\}$, then $\vec{a} + 5.0 = \{6.0, 7.0, 9.0\}$

- **ExtVector** - Element by element addition/subtraction. If one vector has fewer elements than the other, assume the missing elements are zero. For example if $\vec{a} = \{1.0, 2.0, 4.0\}$ and $\vec{b} = \{10.0, 22.0, 30.0, 40.0\}$, then $\vec{a} + \vec{b} = \{11.0, 24.0, 34.0, 40.0\}$
- **Vector** - As described in the previous operation, perform an element by element addition/subtraction of the two vectors.

1.7 Logical Operators

Logical operators must be able to compare a **ExtVector** with another **ExtVector** or a **Vector**. All operations must be **symmetric**.

- **operator==** - Return true if each element of each vector is equivalent. Be careful when comparing double precision values.
- **operator!=** - Return true if each element of each vector is NOT equivalent.

1.8 Index Operator

The `[]` operator should return the element of the **ExtVector** object at a specified index. Assume the vector index starts with one, not zero. If the index is illegal, give an error message and return the value of the *closest* valid element. Note, the user may use this operator to read or **write** to the n^{th} element.

1.9 Output

The insertion operator `<<` should print out the elements of the vector in the following manner. Assume `a` is an **ExtArray** with values `{2.0, 3.5, 4.7}`. If `a` is printed to the screen

```
cout << a << '\n';
```

The output should look like

```
[2, 3.5, 4.7]
```

2 Programming Points

You **must** adhere to all of the following points to receive credit for this program.

1. Turn-in (print-outs and electronically) the files for this program.
2. You must submit the following 8 files (use the names listed below).
 - **ExtVector** must be broken into 3 files
 - `extvec.h` Contains the **ExtVector** class definition.
 - `extvec1.cpp` Contains half of the **ExtVector** member definitions.
 - `extvec2.cpp` Contains remaining half of the **ExtVector** member definitions.
 - `vector.h`, `vector1.cpp`, and `vector2.cpp` Since the **ExtVector** class will work with the **Vector** class, include your **Vector** class files from lab 9 (be certain **Vector** works correctly).
 - `driver.cpp` A *driver* program that tests the **ExtVector** class.
 - `makefile` A makefile to compile the driver program. Note, the makefile must also compile the necessary **Vector** class files and have a `make clean` option.
3. All arrays must be dynamically allocated with **no wasted space!** Therefore, all arrays must be dynamically sized to store **only** the information required. Be certain **no** memory leaks occur.
4. Perform appropriate error checking.