SE106, Fall 2012
Homework 2: Using Template
Assigned: September 14
Due: September 23 24:00

Introduction

In homework 2, you are expected to implement a simplified version of class `Vector`. Details of what interface you should implement will be discussed in detail in the following sections. Since it is a simplified version of `vector`, you do NOT need to implement `iterator`, `alloctor` or thread-safe `mutex`.

Note 1: Unless otherwise mentioned, we do NOT use Stanford library in our homework.

Note 2: You only need to modify `Vector.h`.

Implementation Specification

Methods that you should implement are talked below. You should use `new` and `delete` to manage memory instead of using any other collector class such as `std::vector`.

(default constructor)

```
Vector();
```

Constructs an empty container, with no elements.

(fill constructor)

```
Vector(int size, const T& val);
```

Constructs a container with n elements. Each element is a copy of `val`.

(copy constructor)

```
Vector(const Vector& x);
```

Constructs a container with a copy of each of the elements in x, in the same order.

(destructor)

```
~Vector();
```

Destroys the container object.

operator=

```
Vector<T>& operator=(const Vector<T>& x);
```

Assigns new contents to the container, replacing its current contents, and modifying its size accordingly.

size

```
int size() const;
```

Returns the number of elements in the vector. This is the number of actual objects held in the vector, which is not necessarily equal to its storage capacity.

empty

```
bool empty() const;
```

Returns whether the vector is empty.

operator[]

```
T& operator [](int pos) const;
```

Returns a reference to the element at position n in the vector container. If the requested position is out of range, the behavior is undefined

resize

```
void resize(size_t n, T val);
```

Resizes the container so that it contains n elements.

If n is smaller than the current container size, the content is reduced to its first n elements, removing those beyond (and destroying them).

If n is greater than the current container size, the content is expanded by inserting at the end as many elements as needed to reach a size of n. The new elements are initialized as copies of `val`.

push_back

```
void push_back(const T& val);
```

Adds a new element at the end of the vector, after its current last element. The content of `val` is copied (or moved) to the new element.

Test

Your output should be same as `SampleOutput` file. You can test your homework by using `testVector`. We may use different `testVector` and `runVector.cpp` to test your answers.

Hand in

You only need to turn in your `Vector.h` file.

```
$ turnin homework2@cplusplus Vector.h
```