Data Structures 2024

Homework #1

- (i) Make class *MyStringVector* similar to class *vector<string>* in STL, and also (ii) write a test program to check that all the member functions/operators of your class *MyStringVector* work correctly.

```
class MyStringVector {
        public:
                ...
        private:
                string *str_data;
                ...
};
```

Note that class string is defined in the library <string>.

Constraints

- Use a “pointer to a string” variable and dynamic memory allocation by the *new* operator.
  - string *str_data; // private member
- Do not use any static array!

Members to be Implemented

- Default constructor
  - MyStringVector( );
- Copy constructor for deep copy
  - MyStringVector(const MyStringVector& v);
- Destructor
  - ~MyStringVector( );
- Assignment operator (=) for deep copy
  - Chaining assignment should be possible.

Members to be Implemented

- Operator: +=
  - Appends RHS object to LHS one.

- Operator: [ ]
  - Returns a reference to the string element at the requested position in the vector container.
  - Bound-Check: if the requested position is out of range, it should output some messages and terminate the program.

Members to be Implemented

- void pop_back( );
  - Removes the last string element in the vector, effectively reducing the vector size by one and invalidating all references to it.
- void push_back(string s);
  - Adds a new string element at the end of the vector, after its current last string element. The content of this new string element is initialized to a copy of *s*.
- size_t capacity( ) const;
  - Returns the size of the allocated storage space for the elements of the vector container.

Members to be Implemented

- size_t size( ) const;
  - Returns the number of string elements in the vector container.
- void shrink_to_fit( );
  - Requests the container to reduce its capacity to fit its size.
- void reserve(size_t *n*);
  - Requests that the capacity of the allocated storage space for the string elements of the vector container be at least enough to hold *n* string elements.

Note that size_t is defined in the library <cstdlib>.

Members to be Implemented

- bool is_empty( ) const;
  - Returns whether the vector container is empty, i.e., whether its size is 0.

- void clear( );
  - All the string elements of the vector are dropped: they are removed from the vector container, leaving the container with a size of 0 and a default capacity.

Due Date

- Soft deadline: **Oct. 12, 2024**
- Hard deadline: Oct. 18, 2024

Submission date	Deduction rate
Oct. 13	10 %
Oct. 14	20 %
Oct. 15	30 %
Oct. 16 – Oct. 18	50 %

Notice

- Do not use any container class in STL, such as "vector"!
- Do not use "printf()" and "scanf()" functions!
- You should never use global variables
- Each member function/operator should have its pre-condition and post-condition as comments
  - E.g.,

```
return-type MyStringVector::memberFunction(…);
// precondition: …
// postcondition: …
```

Notice (cont'd)

- Your class will be tested in another test program.
- You should submit a compressed file (**HW1_your-ID.zip**) containing the following four files to the web-site (https://klas.kw.ac.kr/)
  - **HW1_your-ID.hwp**x**/doc**x**/pdf** // report document
  - **HW1_your-ID.cpp/.cc** // your main function (a test program)
  - **MyStringVector.cpp/.cc** // class implementation only
  - **MyStringVector.h** // class documentation & definition only

Notice (cont'd)

- Source code
  - It should be compiled in **Visual Studio 2010 or higher, or g++**
    - **You should note your environment in your report.**
  - Your name and student ID should be noted at the top of your source code in the form of comment
- Report
  - Free format
  - But, it must include several examples of your program and your own discussion
  - It will be an important factor for getting a good score