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( );

pre : 메모리 충분
post : str_data = nullptr
capacity = 0 , size = 0
→ 을 만족하는 vector 생성

- Copy constructor for deep copy
  - MyStringVector(const MyStringVector& v);

pre : other 가 valid 한 상태, 복사가능
post : this 의 private 변수들에
str_data, size, capacity 복사

- Destructor
  - ~MyStringVector( );

pre : constructor가 호출된적 있음.
post : str_data 에 할당한 메모리가 있다면 해제

- Assignment operator (=) for deep copy
  - Chaining assignment should be possible.

pre : ① other 는 valid
② 자기할당 X ← (this == &other 이면 return *this;)
③ 메모리 할당 에러 X

post : ① this는 other 의 size, capacity 복사
② 이전 this 메모리 해제
③ 자기할당 오류시 자신 반환

Members to be Implemented

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

pre:
① other 는 valid
② this 의 capacity는 충분해야한다.
③ ②가 아니라면 확장이 가능해야한다.
④ other.getSize > 0 이면 this 공간 할당 / 확장은 에서 예외 없음.
⑤ & other != this

post:
① this 의 size 는 other의 size.
② other의 모든 요소 this의 뒤에 추가
③ this 객체의 capacity가 필요에 의해 확장
④ this의 유일 객체. other 변화X
⑤ 에외의 할당 필요시 할당

- 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.

pre: index < size
post: index 번째 요소에 대한 참조 반환

Members to be Implemented

pre: vector is not empty
post: (마지막 원소 제거) + (size -= 1)

- void pop_back( );
  - Removes the last string element in the vector, effectively reducing the vector size by one and invalidating all references to it.

pre: vector 가 valid.
post: s가 vector의 마지막에 원소
size += 1
reserve() 호출로 capacity *= 2 될 수 있음.

- 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*.

pre: capacity 가 초기화되어있음.
this가 valid.
post: capacity 를 return. this는 변하지 x

- 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

pre: this가 valid.
post: vector의 빔 여부에 따라 true: false.
vector 변경X

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

pre: this가 valid. data의 size도 valid.
post: size = 0.
데이터는 유지.
capacity 유지. 그 값 재사용 가능.

- 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"! #include <vector> xx
- Do not use "printf()" and "scanf()" functions!
- You should never use global variables C funcl() xx
- 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.hwpx/docx/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