C++ Programming Sections 1762, 1763 & 1764

Deadline December 12, 2022 by 11:59 PM PST

Programming Assignment 6 (minted - November 30, 2022) (200 Points):

OOP & Dynamic Memory Management: A Minimal C++ Container Class with Inheritance

Assignment Description

Build an integer **Container** class defined in namespace **CS52** that has similar behavior to *std::vector*. The **Container** will inherit from an abstract base class named **Sequence**. This assignment uses concepts from Chapters 10,11,12,15 & 16. Topics include user defined types using **classes**, **overloaded operators**, **inheritance**, **virtual functions**, and **abstract classes**.

The integer **Container** will have a default constructor, an overloaded constructor, a copy constructor, and overloaded operators (i.e. bracket [], assignment =, stream insertion << and addition '+'). Use C++'s object oriented programming constructs like classes and resource management that relies on constructors and destructors for memory management. Refer to the interface described on pages 2-3 of this handout for the **Container** class declaration. In A6, the implementations for **Container** and **Sequence** will be separated into interface .h files **Container.h and Sequence.h** and implementation .cpp files **Container.cpp and A06.cpp**, where A06.cpp is the driver/test program.

The **Sequence** base class has three pure virtual functions and a virtual destructor. The code in A6 is organized so that **Container** inherits from **Sequence**. **Container** will *override* the methods **int size(), int capacity()** and **std::string type()** all declared as pure virtual functions in **Sequence**. Note that **type()** will return the *std::string* **"CS52::Container"**. All of the classes will be declared in namespace **CS52**. Refer to page 2 of this handout lists the **Sequence** and **Container** interfaces. Refer to the driver code and output on pages 3-5 in this handout for an example of using the **Container** and polymorphic behavior from using virtual functions.

Where to do the assignment

You can do this assignment on your own computer, or using the SMC Virtual labs. Ensure the code compiles and runs on Windows. Create one zip file named 'A06.zip' with the following files: **Container.h and Sequence.h** and implementation files **Container.cpp and A06.cpp**. Do not use any other name or else points will be deducted.

Submitting the Assignment

Include your name, your student id, the assignment number, the submission date, and a program description in comments at the top of your files, and use the CS52 Programming Guide for coding style guidance. Submit the assignment on Canvas (`https://online.smc.edu`) by **uploading your A06.zip file** to the Assignment 6 entry as an attachment. Do not cut-and-paste your program into a text window. Do not hand in a screenshot of your program's output. Do not hand in a text file containing the output of your program.

Sequence Abstract Class Interface (Sequence.h)

```
/*Sequence class Declaration*/
namespace CS52{
class Sequence
{
public:
        virtual int size() const = 0;
        virtual int capacity() const = 0;
        virtual std::string type() const = 0;
        virtual ~Sequence() {}
};}
```

Container Interface (Container.h)

Container Interface "**is a**" **Sequence** in namespace **CS52**. Override all Sequence virtual functions by implementing the following Container Interface:

```
namespace CS52{
 class Container : public Sequence{
   public:
     Container();                          // Default constructor
     Container( int size, int int_val ); // Overloaded constructor
     Container( const Container& );       // Copy constructor
     ~Container();                         // Destructor

     // Returns a reference to the element at index i
     // throws a string if i is out-of-bounds.
     int& at ( int i ) const throw( std::string );

     // Erases the elements of the Container but does not change capacity.
     void clear();

     // Returns pointer to the first element in the Container.
     int* data() const;

     // If Container is empty return true, else false.
     bool empty() const;

     // Deletes the element at the end of the Container.
     void pop_back();

     // Add an element to the end of the Container.
     void push_back( int element );

     // Returns the number of elements in the Container.
     int size() const override;

     // Returns the allocated storage for the Containter.
     int capacity() const override;

     // Returns the type name ``Container''
         std::string type() const override;

     // Overloaded operators
     int& operator[]     ( int index     );    // [] array syntax
     Container& operator=( const Container& ); // copy assignment

     // overloaded + operator to add two Containers
```

```
     // m = [1,2,3], n=[1,2,3]; o = m+n = [2,4,6]
     //
     // if rhs.size() < this->size()
     // return Container of rhs.size(),
     // otherwise return Container this-size();
     //
     Container operator+(const Container& rhs);

     // Overloaded stream insertion operator
     friend std::ostream& operator<<(std::ostream&, Container&);

         private:
       int   _size     = 0;
       int   _capacity = 0;
       int * _data     = nullptr;
 };//Container
}//namespace
```

Example driver code to test functions in the CS52::Container interface:

```
void info(CS52::Container& c){
    std::cout << "Size is: " << c.size() << "\n";
    std::cout << "Capacity is: " << c.capacity() << "\n";
    std::cout << "Contents : ";
    for (int i = 0; i < c.size(); i++) {
        std::cout << c[i] << " ";
    }
    std::cout << "\n";
}//info

int main() {
    // default constructor, push_back, at methods
    std::cout << "//default constructor, push_back, at \nCS52::Container a;\n";
    CS52::Container a;
    std::cout << "\na.push_back(10); a.push_back(88)\n";
    a.push_back(10); a.push_back(88);
    std::cout << "a.at(0) = 99;\n";
    a.at(0) = 99;
    info(a);

    //b
    std::cout << "\n//overloaded constructor, [] op, at, empty,\
    clear, exception handling: at \nCS52::Container b(2,5);\n";

    CS52::Container b(2,5);
    std::cout << "b is " << b << "\n";

    std::cout << "\n//Add more elements to b\n";
    std::cout << "b.push_back(10); b.push_back(2);\n";
    std::cout << "b.push_back(99); b.push_back(-5);\n";
    b.push_back(10); b.push_back(2);
    b.push_back(99); b.push_back(-5);

    std::cout << "\n//array index [] and at()\n";
    std::cout << "b[0] = 25; b[1] = 1;\n";
    b[0] = 25; b[1] = 1;
    std::cout << "b.at(0) ; b.at(1) ;\n";
```

```
std::cout << b.at(0) << " " << b.at(1) << "\n";

std::cout << "\n\n//empty method, size, and capacity\n";
std::cout << "b.empty() " << (b.empty() ? "True" : "False") << "\n";
std::cout << "b.clear()\n";
b.clear();
info(b);
std::cout << "b.empty() " << (b.empty() ? "True" : "False") << "\n";

std::cout << "\n\n//Exception handling:";

try {
    std::cout << "\nb.at(9) = ";
    std::cout << b.at(9);
}
catch (std::string msg) { std::cerr << "\n" << msg << std::endl; }

//c
std::cout << "\n//copy constructor, copy assignment, pop_back,\
\n//capacity, size\nCS52::Container c(b);\n";

std::cout << "\n//Add more elements to b\n";
std::cout << "b.push_back(11); b.push_back(7);\n";
std::cout << "b.push_back(3); b.push_back(23);\n";
b.push_back(11); b.push_back(7);
b.push_back(3); b.push_back(23);
CS52::Container c(b);
info(c);

//d
std::cout << "\n//copy assignment\n";
CS52::Container d;
d = c;
std::cout << "CS52::Container d = c; " << "\n";
std::cout << "d is " << d << "\n";

std::cout << "\n//size vs capacity()\n";
std::cout << "d.size() is "     << d.size() << "\n";
std::cout << "d.capacity() is " << d.capacity() << "\n";

std::cout << "\n//pop_back()\n";
std::cout << "d.pop_back();d.pop_back();\n";
d.pop_back(); d.pop_back();

std::cout << "\n//size vs capacity\n";
std::cout << "d.size() is " << d.size() << "\n";
std::cout << "d.capacity() is " << d.capacity() << "\n";

//Test overloaded + operator
CS52::Container m(5, 3);
CS52::Container n(3, 0);
n[0] = 55; n[1] = 100; n[2] = 500;

CS52::Container o = m + n;

    std::cout << "Test overloaded + operator\n";
    std::cout << "m = [3,3,3,3,3], n = [55,100,500]\n";
```

```
        std::cout << "o = m + n = [" << o << "]\n";

    const int SIZE = 2;

    CS52::Sequence* sequence[SIZE];

    sequence[0] = new CS52::Container(2, 1);
    sequence[1] = new CS52::Container(5, 3);

    // Print out the type and length of each Sequence element
    std::cout << "\nTest polymorphism \n";

    for (int i = 0; i < SIZE; i++)
    {
        std::cout << "Sequence [" << i << "]" << " is a " << sequence[i]->type() \
        << ": " << *(dynamic_cast<CS52::Container*> (sequence[i])) << "\n";
    }

    for (int i = 0; i < 2; i++)
        delete sequence[i];

    char stop; std::cin >> stop; return 0;

}//main -- end of example driver code
```

Example test output from code in main above that uses our Container class:

```
//default constructor, push_back, at
CS52::Container a;

a.push_back(10); a.push_back(88)
a.at(0) = 99;
Size is: 2
Capacity is: 2
Contents : 99 88

//overloaded constructor, [] op, at, empty, clear, exception handling: at
CS52::Container b(2,5);
b is 5 5

//Add more elements to b
b.push_back(10); b.push_back(2);
b.push_back(99); b.push_back(-5);

//array index [] and at()
b[0] = 25; b[1] = 1;
b.at(0) ; b.at(1) ;
25 1

//empty method, size, and capacity
b.empty() False
b.clear()
Size is: 0
Capacity is: 8
Contents :
b.empty() True
```

```
//Exception handling:
b.at(9) =
out-of-bounds

//copy constructor, copy assignment,
//pop_back, capacity, size
CS52::Container c(b);

//Add more elements to b
b.push_back(11); b.push_back(7);
b.push_back(3); b.push_back(23);
Size is: 4
Capacity is: 8
Contents : 11 7 3 23

//copy assignment
CS52::Container d = c;
d is 11 7 3 23

//size vs capacity()
d.size() is 4
d.capacity() is 8

//pop_back()
d.pop_back();d.pop_back();

//size vs capacity
d.size() is 2
d.capacity() is 8

Test overloaded + operator
m = [3,3,3,3,3], n = [55,100,500]
o = m + n = [58 103 503]

Test polymorphism
Sequence [0] is a CS52::Container: 1 1
Sequence [1] is a CS52::Container: 3 3 3 3 3
```

Saving your work
Save your work often on a flash-drive or to the cloud (e.g., GoogleDrive, Microsoft OneDrive, Canvas, etc.). Always save a personal copy of your files (e.g. .cpp, .h, etc.). Do not store files on the virtual lab computers.

Do your own work

Do not distribute this handout. Do not upload to chegg, coursehero, ankitcodinghub, or any other online platform. Do not pay someone to write the code and submit their code as your solution. You are expected to do your own work. Turning in code that is not your own work will result in a referral to Student Judicial Affairs.

Assignment 6 References:

- Refer to **string class** in Sec. 11.4 of textbook for dynamic memory & OOP resource management.
- MSDN STL vector docs and cplusplus.com vector docs
- resize docs at cppreference.com and resize docs at microsoft
- reserve docs at cppreference.com and reserve docs at microsoft