Khoi Duong

Prof. Yang

CS360L

6/28/2022

1.

Source code:

```cpp
#include <iostream>      // std::cout
#include <algorithm>     // std::sort
#include <vector>        // std::vector
int main (void) {
  int arr[] = {3,7,12,45,26,81,52,33};
//relying on class container sort function to implement sorting for a given array.
    std::sort(arr, arr+8);
    std::cout << "Sorted array: ";
    for (int i = 0; i < 8; i++)
        std::cout << arr[i] << " ";
    std::cout << std::endl;
    return 0;
}
```

Run program & result:

```
PS D:\VS CODE\C C++\CS360\Midterm> cd "d:\VS CODE\C C++\CS360L\Midte
rm\" ; if ($?) { g++ 1.cpp -o 1 } ; if ($?) { .\1 }
Sorted array: 3 7 12 26 33 45 52 81
```

2.

Source code:

```cpp
// Create a class A with a private static variable x, initialize x, and increase 1
to x in a method
// within the class A file. after that, verify it in the main function

#include <iostream>
using namespace std;
```

```
class A {
    private:
    static int x;

    public:
    // initialize x
    A() {
        cout << "x is " << x << endl;
    }

    // increase x by 1 in a method within the class A file
    void increase() {
        cout << "Increasing x by 1" << endl;
        x++;
    }

    int output() {
        return x;
    }
};

int A::x = 10;
// verify in the main function
int main() {
    A a;
    a.increase();
    cout << "x is " << a.output() << endl;
    return 0;
}
```

Run program & result:

```
x is 10
Increasing x by 1
x is 11
```

3.

Source code:

```cpp
#include <iostream>
using namespace std;

class A{
    int x = 10;
    friend class B;
    // The code below will produce errors since class A is not a friend of class B
    // void output(B b){
    //     cout << "y is " << y << endl;
    // }
};

class B{
    int y;
    public:
    void output(A a){
        cout << "x is " << a.x << endl;
    }
};

int main(){
    A a;
    B b;
    b.output(a);
    return 0;
}
```

Run program & result:

```
x is 10
```

4.

Source code:

```cpp
// Given class A & class B, and an object b of B in class A
// write a program to initialize b in class A by the constructor with some arguments
that you decide
#include <iostream>
using namespace std;

class B{
    public:
    int y;
```

```
    B(int y){
        this->y = y;
    }
};

class A: public B{
    int x;
    public:
    // object b of class B in class A
    A(int y): B(y){};
};

int main(){
    A a(20);
    cout << "y = " << a.y << endl;
    return 0;
}
```

Run program & result:

```
y = 20
```

5.

The “int const*” and “const int*” is a pointer to constant integer. This means p and q are pointers to the constant value that should not be changed. The “const” qualifier does not affect the pointer, therefore both “int const*” and “const int*” are the same.

6.

Source code:

```
#include <iostream>
using namespace std;


class HugeInteger{
  public:
    int digits[50];
    // constructor
```

```
HugeInteger(){
  for(int i = 0; i < 40; i++){
    digits[i] = 0;
  }
}

void input(){
        string n;
        cout << "Enter a huge integer: ";
        cin >> n;
        for(int i = 0; i < n.length() ; i++){
            int a = stoi(n.substr((n.length()-1-i), 1));
            digits[i] = a;
        }
    }

void output(){
        for(int i = 39; i >= 0; i--){
            // pass zero located to the left of the first non-zero digit
            if(digits[i] != 0){
                for(int j = i; j >= 0; j--){
                    cout << digits[j];
                }
                break;
            }
            if(digits[i] == 0 && i == 0){
                cout << 0;
            }
        }
        cout << endl;
    }

// overload operator+ for HugeInteger
HugeInteger operator+(int rhs){
    HugeInteger result;
    int array[50];
    for(int i = 0; i < 50; i++){
        array[i] = rhs % 10;
        rhs /= 10;
    }
    for(int i = 0; i < 50; i++){
        int sum = this->digits[i] + array[i] + result.digits[i];
        if(sum >= 10){
            result.digits[i+1] += 1;
            sum -= 10;
```

```
            }
            result.digits[i] += sum;
        }
        return result;
    }

};

int main(void){
  HugeInteger bigInteger;
  bigInteger.input();
  int integer;
  cout << "Enter an integer: ";
  cin >> integer;
  bigInteger = bigInteger + integer;
  cout << "The sum of the two numbers is: ";
  bigInteger.output();
  return 0;
}
```

Run program & result:

```
Enter a huge integer: 6519845654
Enter an integer: 65165
The sum of the two numbers is: 65191020919
```

7.

A::operator otherClass();

Source code:

```
#include <iostream>
using namespace std;

class A{
    // implement user-define casting operator to covert a class A to the other class
    public:
    A(int x){
        this->x = x;
    }
    int x;
    operator int() const {
        return 7;
```

```
    }
};

int main(){
    A a(20);
    cout << "a = " << a << endl;
    int m = a;
    cout << "m = " << m << endl;
    return 0;
}
```

Run program & result:

```
a = 7
m = 7
```