1.
a. This one will NOT generate error but declaring a variable (pointer or not) without initializing will assign garbage value to the variable and may cause logic error later in the program.

b. This will generate compile time error as the two pointers have different types (long vs double). To correct this error, we need to (1) change one of the two to match the type (either both long or both double) or (2) change one of them to be *void or (3) cast the type in the assignment e.g. integerPtr = (int*)realPtr; .

c. This will generate compile time error since x is int pointer whereas y is int and we cannot assignment int to int* variable. We can declare both as int or both as int* e.g. int *x, *y; .

d. Compile time error will be generated as s is declared built-in array, to correct this, we should declare s as pointer (built-in array is essentially pointer)
char *s{“this is a char array”}; .

e. Compile time error since genericPtr is void* and cannot be used directly in the assignment by dereferencing. We can either declare the genericPtr as int* or in the assignment we cast it to int pointer.

f. xPtr is not declared as pointer as is initialized with the address & notation, this will generate compile time error, we should declare xPtr this way: double *xPtr{&x}; .

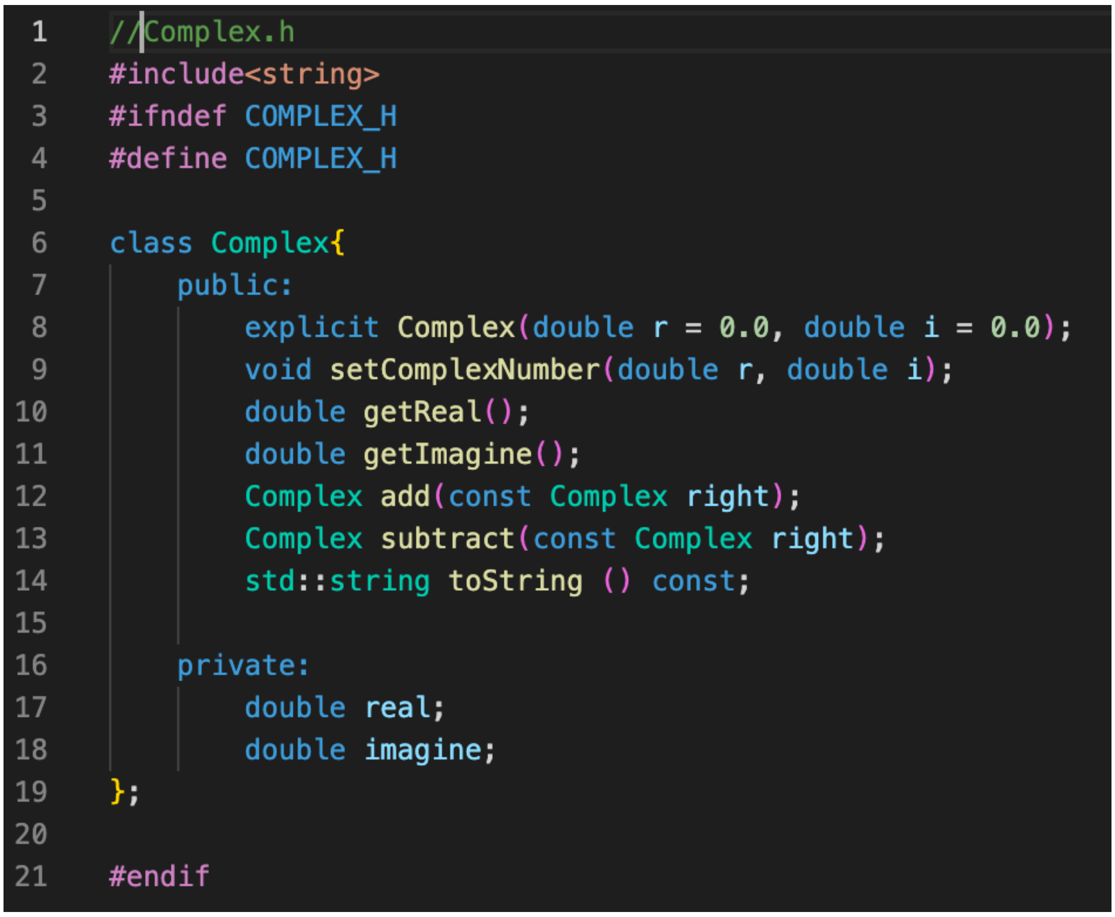
2.

The function mystery1 is essentially concatenating two strings, or more precisely, it appends s2 to s1. It first loop through s1 and make the pointer pointing to the end of the string '\0' and then uses a for loop to assign each char of s2 to the end of s1 one by one. And the main program asks for input for two strings then concat them and eventually print the combined string. For example, if s1 is entered as "aaa" and s2 is "bbb" then the main program will print "aaabbb".

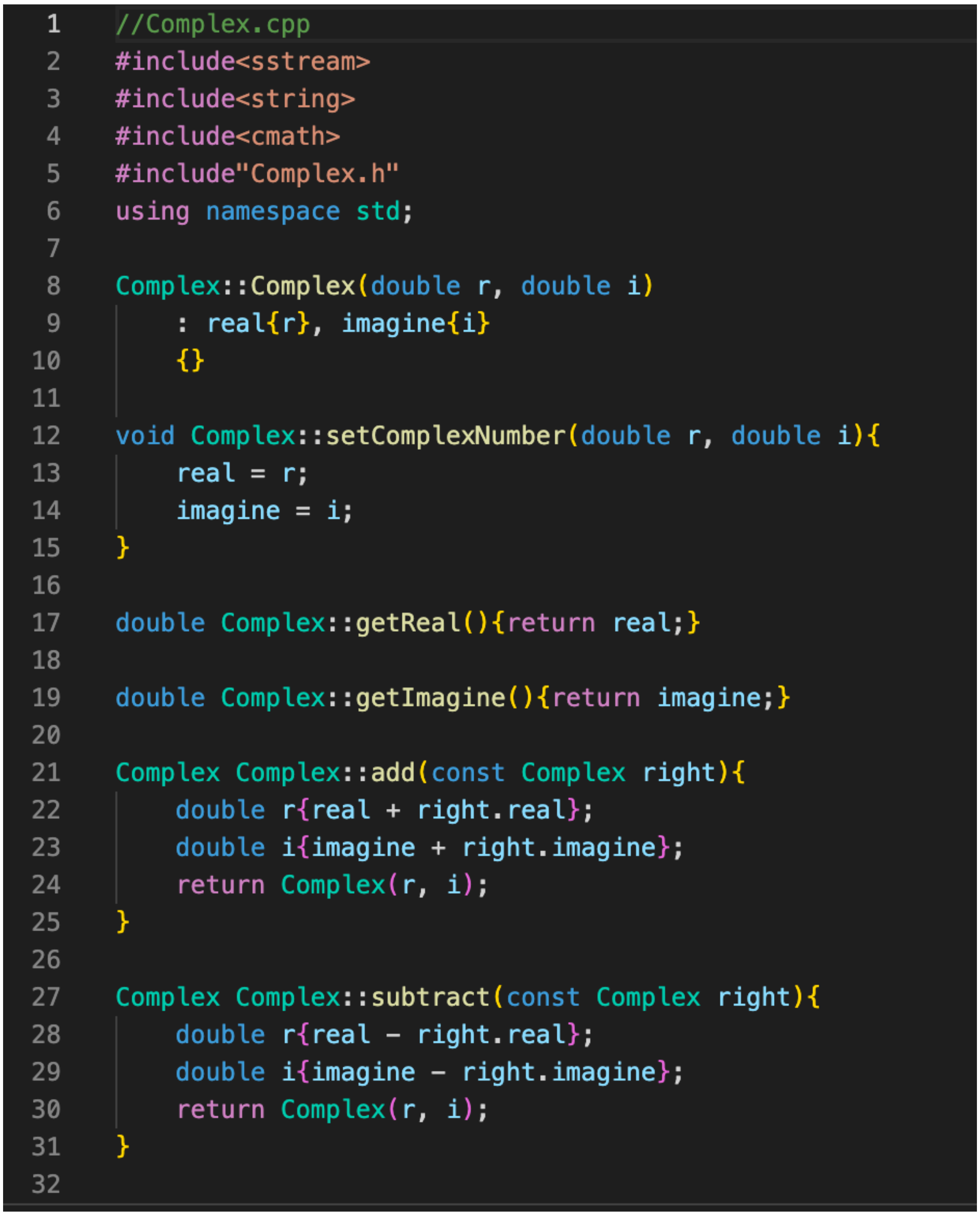
3.

The function mystery2 will count how many chars are there in the string passed to the function and return the number which is the length of the string. For example, if the string passed is "abc", the function will return 3 and 3 will be printed in the main function.

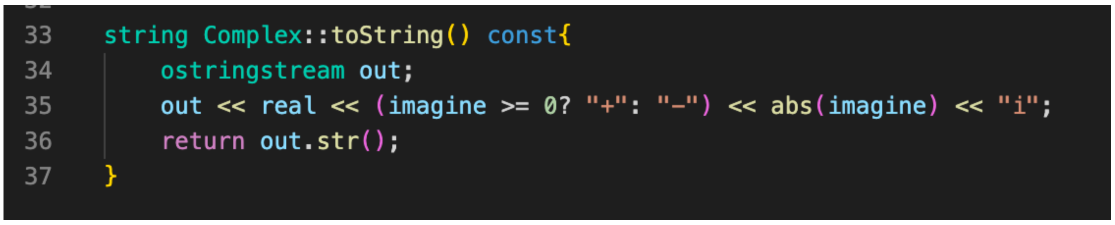
4.



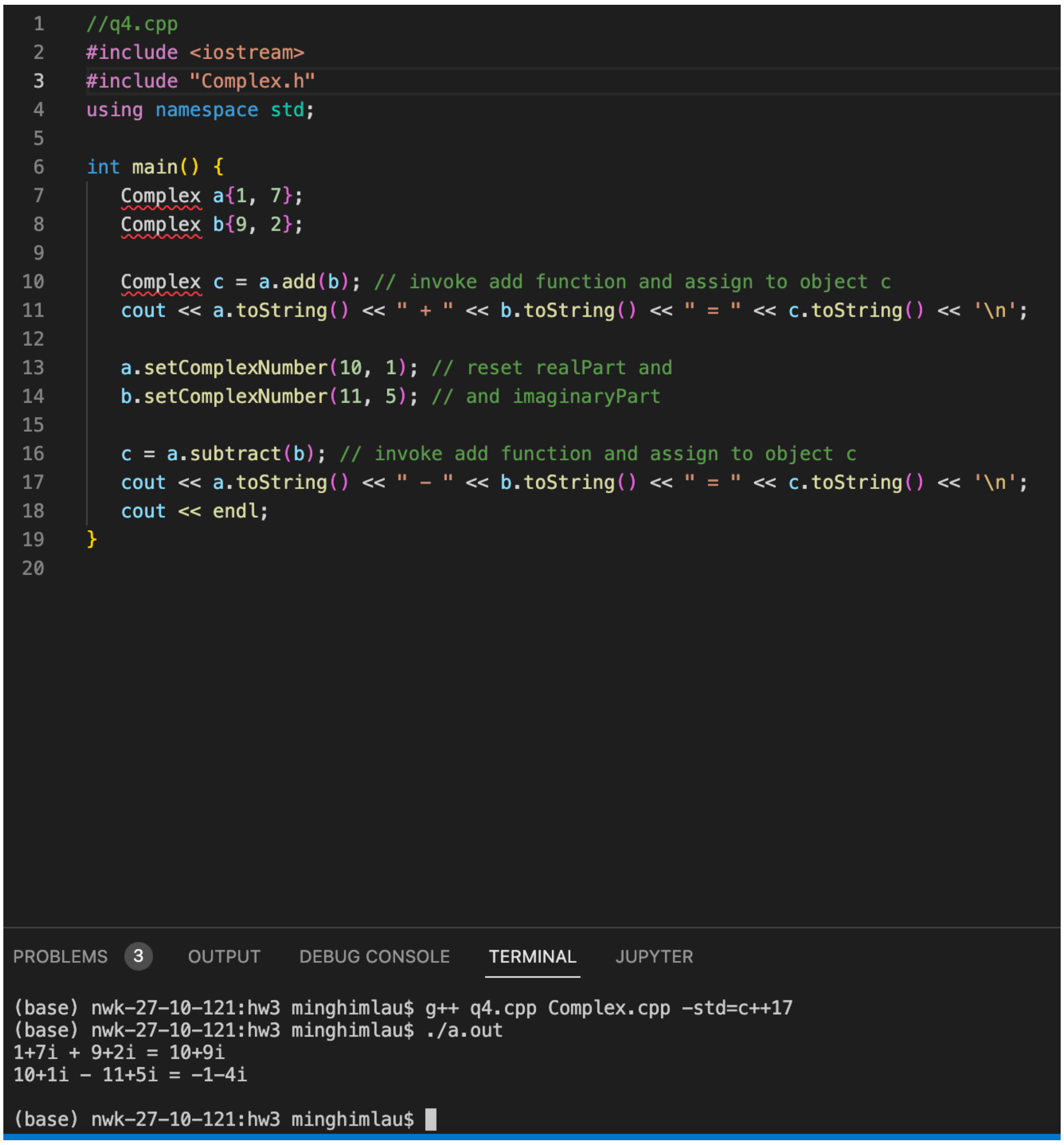
```cpp
//Complex.h
#include<string>
#ifndef COMPLEX_H
#define COMPLEX_H

class Complex{
    public:
        explicit Complex(double r = 0.0, double i = 0.0);
        void setComplexNumber(double r, double i);
        double getReal();
        double getImagine();
        Complex add(const Complex right);
        Complex subtract(const Complex right);
        std::string toString () const;

    private:
        double real;
        double imagine;
};

#endif
```



```cpp
//Complex.cpp
#include<sstream>
#include<string>
#include<cmath>
#include"Complex.h"
using namespace std;

Complex::Complex(double r, double i)
    : real{r}, imagine{i}
    {}

void Complex::setComplexNumber(double r, double i){
    real = r;
    imagine = i;
}

double Complex::getReal(){return real;}

double Complex::getImagine(){return imagine;}

Complex Complex::add(const Complex right){
    double r{real + right.real};
    double i{imagine + right.imagine};
    return Complex(r, i);
}

Complex Complex::subtract(const Complex right){
    double r{real - right.real};
    double i{imagine - right.imagine};
    return Complex(r, i);
}

```



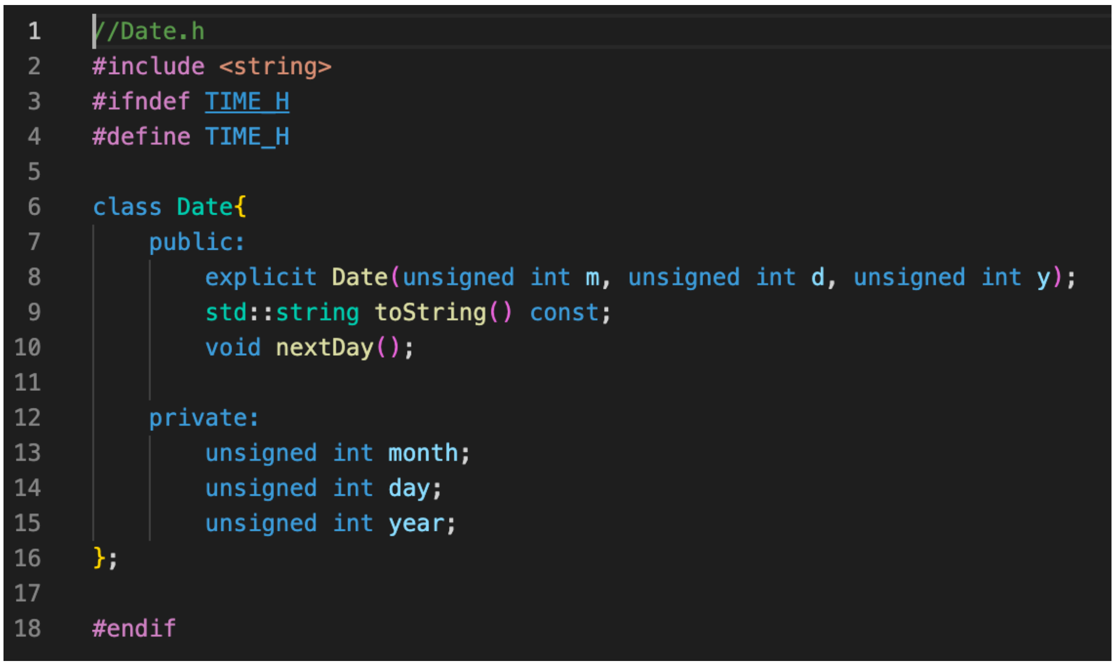
```cpp
string Complex::toString() const{
    ostringstream out;
    out << real << (imagine >= 0? "+": "-") << abs(imagine) << "i";
    return out.str();
}
```



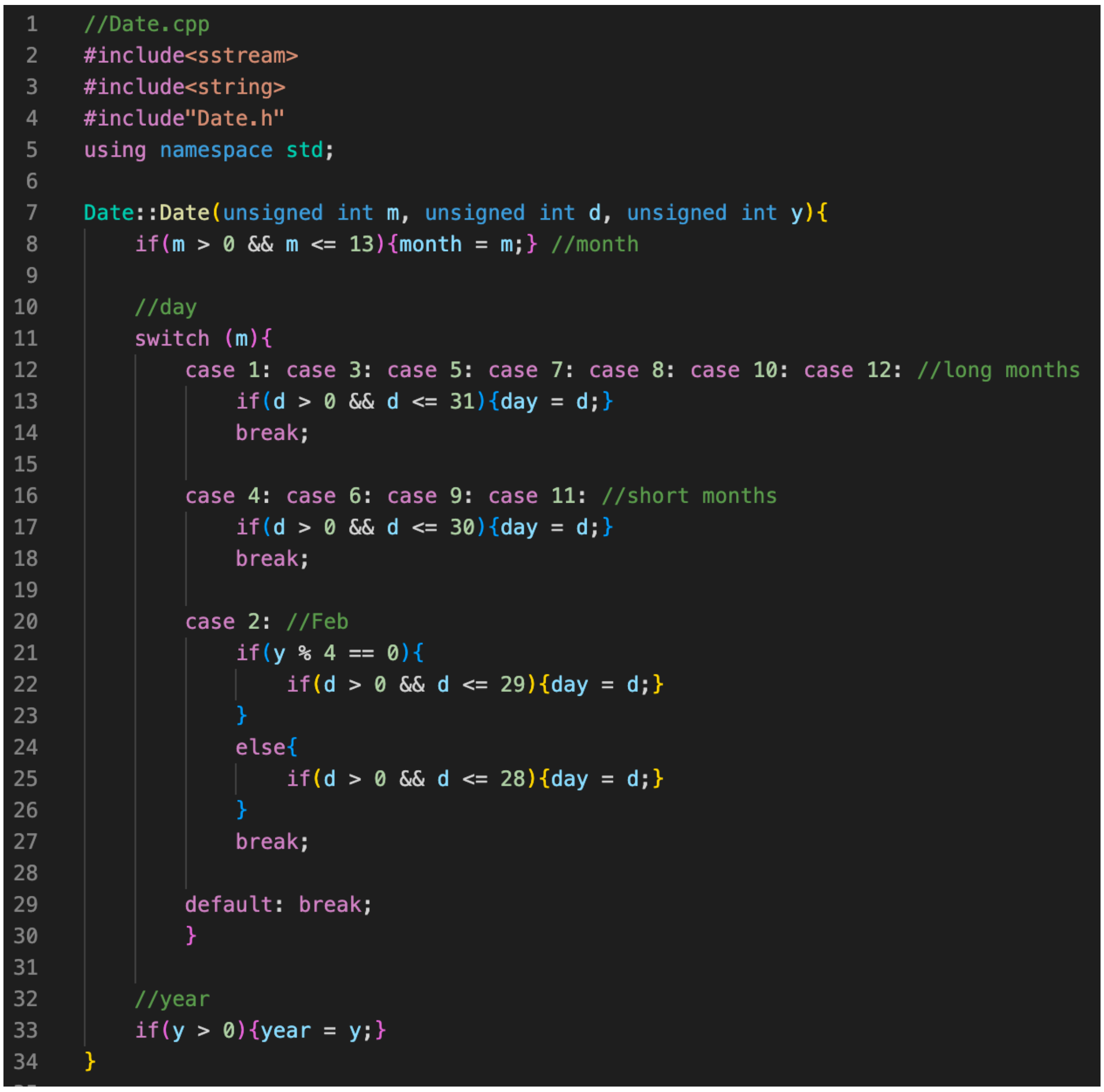
```cpp
//q4.cpp
#include <iostream>
#include "Complex.h"
using namespace std;

int main() {
    Complex a{1, 7};
    Complex b{9, 2};

    Complex c = a.add(b); // invoke add function and assign to object c
    cout << a.toString() << " + " << b.toString() << " = " << c.toString() << '\n';

    a.setComplexNumber(10, 1); // reset realPart and
    b.setComplexNumber(11, 5); // and imaginaryPart

    c = a.subtract(b); // invoke add function and assign to object c
    cout << a.toString() << " - " << b.toString() << " = " << c.toString() << '\n';
    cout << endl;
}

```

```
(base) nwk-27-10-121:hw3 minghimlau$ g++ q4.cpp Complex.cpp -std=c++17
(base) nwk-27-10-121:hw3 minghimlau$ ./a.out
1+7i + 9+2i = 10+9i
10+1i - 11+5i = -1-4i

(base) nwk-27-10-121:hw3 minghimlau$ 
```

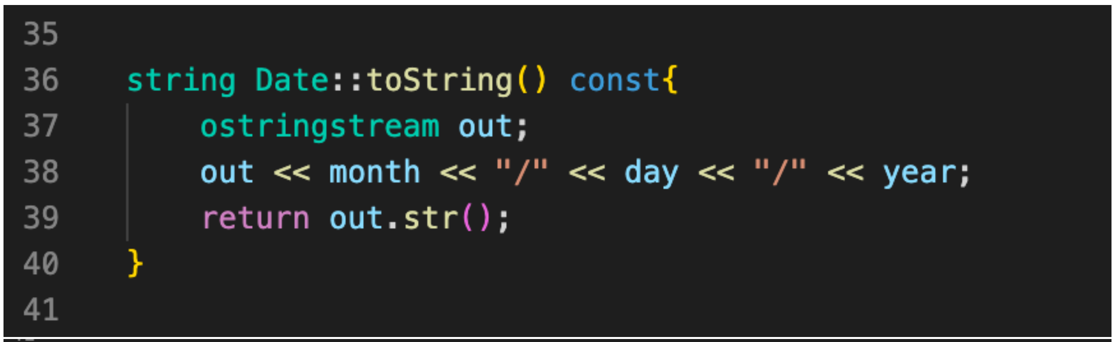
5.



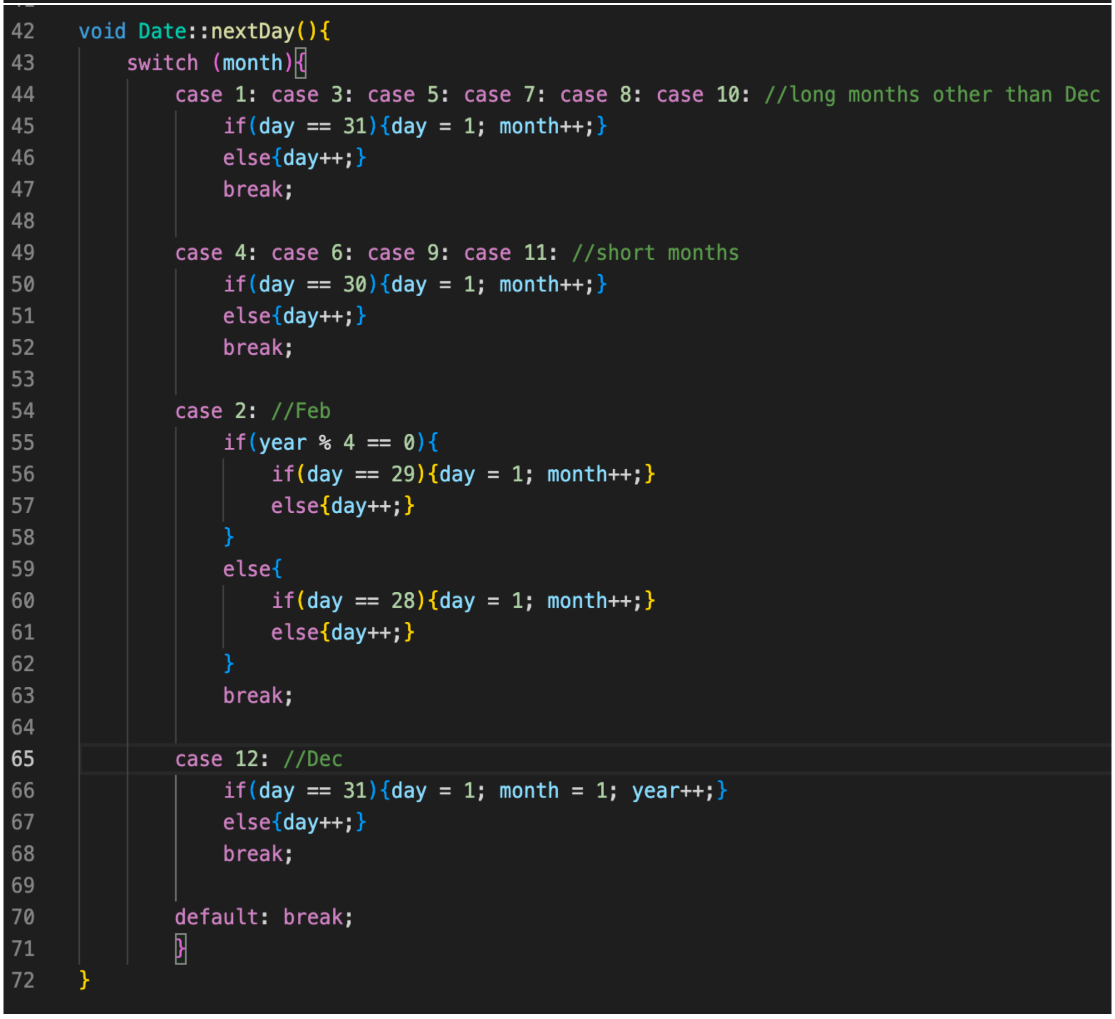
```cpp
//Date.h
#include <string>
#ifndef TIME_H
#define TIME_H

class Date{
    public:
        explicit Date(unsigned int m, unsigned int d, unsigned int y);
        std::string toString() const;
        void nextDay();

    private:
        unsigned int month;
        unsigned int day;
        unsigned int year;
};

#endif
```



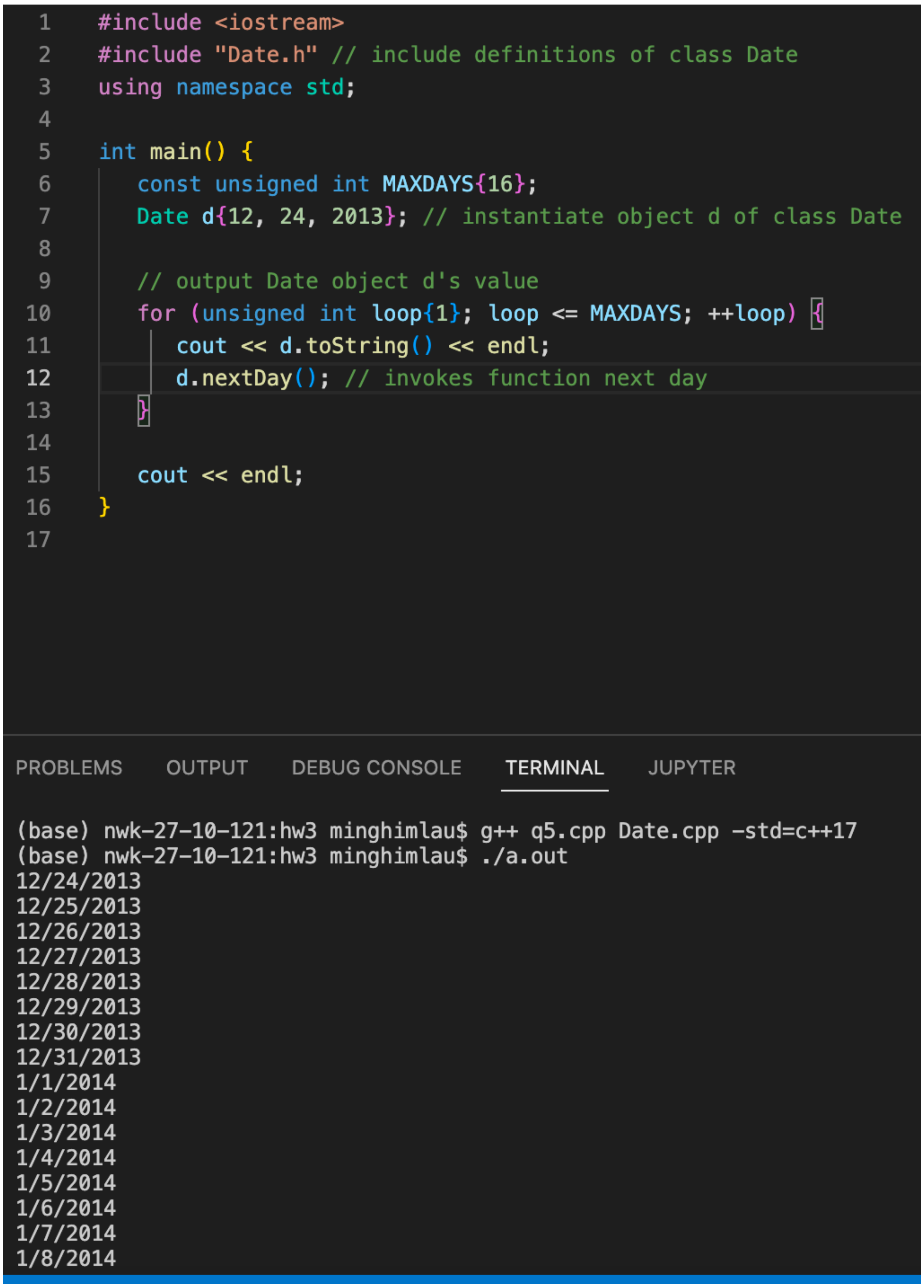
```cpp
//Date.cpp
#include<sstream>
#include<string>
#include"Date.h"
using namespace std;

Date::Date(unsigned int m, unsigned int d, unsigned int y){
    if(m > 0 && m <= 13){month = m;} //month

    //day
    switch (m){
        case 1: case 3: case 5: case 7: case 8: case 10: case 12: //long months
            if(d > 0 && d <= 31){day = d;}
            break;

        case 4: case 6: case 9: case 11: //short months
            if(d > 0 && d <= 30){day = d;}
            break;

        case 2: //Feb
            if(y % 4 == 0){
                if(d > 0 && d <= 29){day = d;}
            }
            else{
                if(d > 0 && d <= 28){day = d;}
            }
            break;

        default: break;
        }

    //year
    if(y > 0){year = y;}
}
```



```cpp

string Date::toString() const{
    ostringstream out;
    out << month << "/" << day << "/" << year;
    return out.str();
}

```



```cpp
void Date::nextDay(){
    switch (month){
        case 1: case 3: case 5: case 7: case 8: case 10: //long months other than Dec
            if(day == 31){day = 1; month++;}
            else{day++;}
            break;

        case 4: case 6: case 9: case 11: //short months
            if(day == 30){day = 1; month++;}
            else{day++;}
            break;

        case 2: //Feb
            if(year % 4 == 0){
                if(day == 29){day = 1; month++;}
                else{day++;}
            }
            else{
                if(day == 28){day = 1; month++;}
                else{day++;}
            }
            break;

        case 12: //Dec
            if(day == 31){day = 1; month = 1; year++;}
            else{day++;}
            break;

        default: break;
        }
}
```



```cpp
#include <iostream>
#include "Date.h" // include definitions of class Date
using namespace std;

int main() {
    const unsigned int MAXDAYS{16};
    Date d{12, 24, 2013}; // instantiate object d of class Date

    // output Date object d's value
    for (unsigned int loop{1}; loop <= MAXDAYS; ++loop) {
        cout << d.toString() << endl;
        d.nextDay(); // invokes function next day
    }

    cout << endl;
}
```

```
(base) nwk-27-10-121:hw3 minghimlau$ g++ q5.cpp Date.cpp -std=c++17
(base) nwk-27-10-121:hw3 minghimlau$ ./a.out
12/24/2013
12/25/2013
12/26/2013
12/27/2013
12/28/2013
12/29/2013
12/30/2013
12/31/2013
1/1/2014
1/2/2014
1/3/2014
1/4/2014
1/5/2014
1/6/2014
1/7/2014
1/8/2014
```