```
#include"Binary.h"
#include<bitset>
#include<cmath>

Binary::Binary() : Roman()
{
	int_to_Binary();
}

Binary::Binary(int n) : Roman(n)
{
	int_to_Binary();
}

void Binary::int_to_Binary()
{
	binary = bitset<16>(get_number()).to_string();
}

string Binary::get_Binary_style()
{
	int_to_Binary();
	return binary;
}
```

```
#ifndef BINARY_H
#define BINARY_H

#include"roman.h"

class Binary : public Roman
{
private:
        string binary;
        void int_to_Binary();      // converting int to string Roman Numeral
public:
        Binary();
        Binary(int n);
        string get_Binary_style();
};
#endif



#include"Number.h"
#include<iostream>
#include<iomanip>
#include<locale>
#include<algorithm>
#include<sstream>

Number::Number()
```

```
{
	number = 0;
	int_to_EURO();
	int_to_US();
}
Number::Number(int n)
{
	number = n;
	int_to_US();
	int_to_EURO();
}

void Number::int_to_US()
{
	stringstream ss;
	ss.imbue(locale(""));
	ss << fixed << number;
	US = ss.str();
}

void Number::int_to_EURO()
{
	stringstream ss;
	ss.imbue(locale(""));
	ss << fixed << number;
	EURO = ss.str();
	replace(EURO.begin(), EURO.end(), ',', '.');
```

```
}
string Number::get_US_style() const
{
	return US;
}

// Returns int as a string in Euro style.
string Number::get_EURO_style() const
{
	return EURO;
}

// Returns the int.
int Number::get_number() const
{
	return number;
}

// Sets the value of int, US, and EURO.
void Number::set_number(int n)
{
	number = n;
	int_to_US();
	int_to_EURO();
}
```

```
#ifndef NUMBER_H
#define NUMBER_H

#include<string>
using namespace std;

class Number
{
public:
        // Default constructor without any initial value.
        Number();

        // Constructor with initial value.
        Number(int);

        // Returns int as a string in US style.
        string get_US_style()const;

        // Returns int as a string in Euro style.
        string get_EURO_style()const;

        // Returns the int.
        int get_number()const;

        // Sets the value of int, US, and EURO.
        void set_number(int);
```

```
private:
        // A decimal number less than 4000.
        int number;

        // A string represent number in US style.
        string US;

        // A string representing number in EURO style.
        string EURO;

        // converting int to string in US style
        void int_to_US();

        // converting int to string in EURO style
        void int_to_EURO();
};

#endif
```



```
#include"roman.h"
Roman::Roman() : Number()
{
        roman = "INVALID";
}
```

```
Roman::Roman(int n) :Number(n)
{
	int_to_Roman();
}

void Roman::int_to_Roman()
{
	int n = get_number();
	roman = "";

	if (n <= 0)
	{
		roman = "INVALID";
	}
	else
	{
		int th, h, t, o;
		string thousands[] = { "", "M", "MM", "MMM", "MMMM" };
		string hundreds[] = { "", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM" };
		string tens[] = { "", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC" };
		string ones[] = { "", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX" };
		th = n / 1000;
		n = n % 1000;
		h = n / 100;
		n = n % 100;
		t = n / 10;
```

```
                o = n % 10;
                roman += thousands[th];
                roman += hundreds[h];
                roman += tens[t];
                roman += ones[o];
        }
}
string Roman::get_ROMAN_style()
{
        int_to_Roman();
        return roman;
}
```



```
#ifndef ROMAN_H
#define ROMAN_H

#include"Number.h"

class Roman: public Number
{
private:
        string roman;
        void int_to_Roman();    // converting int to string Roman Numeral
public:
        Roman();
        Roman(int n);
```

```
        string get_ROMAN_style();
};
#endif
```

```
#include"Binary.h"
#include<Windows.h>
#include<iostream>
#include<cassert>
int main()
{
        string output = "";
        do
        {
                int ch;
                cout << "Enter 1 if you want to change from decimal to Roman numeral.\n";
                cout << "Enter 2 if you want to change from decimal to Binary numeral.\n";
                cout << "Enter 3 if you want to display decimal in US style.\n";
                cout << "Enter 4 if you want to display decimal in Euro style.\n";
                cout << "Enter a non - digit to quit.\n";
                cin >> ch;
                assert(ch >= 0 && "No valid input was entered.\nProgram Terminating...");
                assert(ch <= 4 && "No valid input was entered.\nProgram Terminating...");
                if (cin.fail())
                {
                        system("cls");
                        cout << output;
```

```
            system("pause");
            return 0;
        }
        else
        {
            int number;
            cout << "Enter a positive integer smaller than 4000:  ";
            cin >> number;
            assert(number >= 0 && "Negative number was entered.\nProgram
Terminating...");
            assert(number <= 4000 && "Number was greater than 4000.\nProgram
Terminating...");
            Binary n(number);
            switch (ch)
            {
            case 1:
                cout << n.get_ROMAN_style() << endl;
                output += n.get_ROMAN_style();
                output += "\n";
                break;

            case 2:
                cout << n.get_Binary_style() << endl;
                output += n.get_Binary_style();
                output += "\n";
                break;

            case 3:
```

```
                cout << n.get_US_style() << endl;
                output += n.get_US_style();
                output += "\n";
                break;

            case 4:
                cout << n.get_EURO_style() << endl;
                output += n.get_EURO_style();
                output += "\n";
                break;
            default:
                break;
            }
        }
        Sleep(3000);
        system("cls");
    } while (1);
}

class number
{
protected:
{
int ch,a;
long US,EURO;
}
}
```

```
class rom:public number
{
public:
void binary()
{
cout<<"enter value"|<<endl;
cin>>a;
int remainder;
if(a<=1)
{
cout<<number;
return;
}
remainder=a%2;
cout<<remainder;
}
void usstyle()
{
cout<<"enter the value"<<endl;
cin>>a;
while(a>0)
{
a=a/10;
int i++;
}
if(i>3)
{
```

```
int b=a/1000;
a%=1000;
cout<<"US="<<b<<","<<a<<endl;
}
else
{
cout<<a;
}
}
void eurostyle()
{
cout<<"enter the numbers";
cin>>a;
while(a>0)
{
a=a/10;
int i++;
}
if(i>3)
{
int b=a/1000;
a%=1000;
cout<<"EURO="<<c<<","<<b<<endl;
}
else
{
cout<<a;
```

```
}
}
void main()
{
rome r;
cout<<"1.binary\t2.US\t3.EURO";
switch(ch)
{
case 1: r.binary();
      break;
case 2: r.usstyle();
      break;
case 3: r.eurostyle();
      break;
}
}


class number
{
protected:
{
int ch,a;
long US,EURO;
}
}
class rom:public number
```

```
{
public:
void binary()
{
cout<<"enter value"|<<endl;
cin>>a;
int remainder;
if(a<=1)
{
cout<<number;
return;
}
remainder=a%2;
cout<<remainder;
}
void usstyle()
{
cout<<"enter the value"<<endl;
cin>>a;
while(a>0)
{
a=a/10;
int i++;
}
if(i>3)
{
int b=a/1000;
```

```
a%=1000;
cout<<"US="<<b<<","<<a<<endl;
}
else
{
cout<<a;
}
}
void eurostyle()
{
cout<<"enter the numbers";
cin>>a;
while(a>0)
{
a=a/10;
int i++;
}
if(i>3)
{
int b=a/1000;
a%=1000;
cout<<"EURO="<<c<<","<<b<<endl;
}
else
{
cout<<a;
}
```

```
}
void main()
{
rome r;
cout<<"1.binary\t2.US\t3.EURO";
switch(ch)
{
case 1: r.binary();
      break;
case 2: r.usstyle();
      break;
case 3: r.eurostyle();
      break;
}
}
```