

1. Read the codes carefully and answer the following questions.

```
#include<iostream>
using namespace std;
class Singleton
{
private:
    static Singleton *s;
    Singleton() { cout << "Constructor" << endl; }
public:
    static Singleton* GetInstance()
    {
        if (s == nullptr)    s = new Singleton();
        return s;
    }
    ~Singleton()
    {
        if (s != nullptr)
        {
            delete s;
            cout << "Release the static s." << endl;
        }
    }
};
Singleton* Singleton::s = nullptr;
int main()
{
    Singleton *ps;
    ps = Singleton::GetInstance();
    cout << ps << endl;

    return 0;
}
```

- 1.1 Please explain the member variable s;
- 1.2 Please describe the meaning above the codes;
- 1.3 Correct errors you find if any. Don't modify any codes in MAIN.

2. Create a class, CMyStack, to save double-type elements and implement the operations on the stack as follow members:

```
class CMyStack
{
private:
    double *m_pTop;        // Top pointer of stack
    int m_iSize;           // Number of actual elements
    int m_iCapacity;       // Capacity of stack
}
```

```

public:
    CMyStack(int size);    // Set the capacity of stack
    ~CMyStack();
    double Pop();
    double Peek();
    bool Push(double ch);
    bool isEmpty();        // Is Stack empty?
    bool isFull();         // Is Stack full?
    int  GetSize();        // Get number of actual elements
    void Clear();          // Clear stack
};

```

Note: Don't modify any member variables and interface of member function in CMyStack.

3. Create a class , CExpression, to calculate the value of an expression which consists of numbers and operators such as +, -, \*, / and ( ).

Define member functions such as following:

```

class CExpression
{
private:
    // .....
public:
    double Value( );    // Get value of expresstion
    ostream& operator << (ostream& os, const CExpression& expr);    // print only
expression except its value
    // .....
};

```

NOTE:

3.1 You can define appropriate member functions and variables.

3.2 You MUST use CMyStack you have finished to complete the program together.

3.2 Assume that an expression you input is always correct, that is , there is no grammatical errors.

CExpression can be used in the following way in the main:

```

int main()
{
    CExpression expr("50.3-20.12+8*8/2");
    cout << expr << " = " << expr.Value() << endl;    // 50.3-20.12+8*8/2 = 62.18

    expr.SetExpression("55.99-88.11+77.12");
    cout << expr << " = " << expr.Value() << endl;    // 55.99-88.11+77.12 = 45
}

```

```

expr.SetExpression("(39+11)*30+10/5");
cout << expr << " = " << expr.Value() << endl;    // (39+11)*30+10/5 = 1502

expr.SetExpression("39+12*(47+33)");
cout << expr << " = " << expr.Value() << endl;    // 39+12*(47+33) = 999

expr.SetExpression("20/(112-(10*1.2))/10-1.01");
cout << expr << " = " << expr.Value() << endl;    // 20/(112-(10*1.2))/10-1.01 = -0.99

expr.SetExpression("20/(112-10*1.2)/10-1.01");
cou << exprt << " = " << expr.Value() << endl;    // 20/(112-10*1.2)/10-1.01 = -0.99

cout << "ENDING..." << endl;
return 0;
}

```

[optional] Create a class, CLINT, to save a big positive integer which is no more than 100 digits. Define a member function to achieve the sum of two big numbers such as following:

```

class CLINT
{
private:
    // .....
public:
    CLINT Add(const CLINT& L);    // Achieve the sum of two big numbers
    void Value();                // Display the big number
    // .....
};

```

CLINT can be used in the following way in the main:

```

int main()
{
    LINT L1("12345678900987654321"), L2("9876543210"), L3;
    L3 = L1.Add(L2);
    L3.Value();    // 12345678910864197531

    return 0;
}

```

NOTES: You can define appropriate member functions and variables.