

公告

这是期末考试的第一部分，在这部分答题时，不能开启考试监考

📄 填空题 15

💻 程序填空题 2

4-1 The output of the code below is:

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

class Str{
    char m_s[10];
    char *m_p;
public:
    Str(char *s){strcpy(m_s,s);m_p = m_s;}
    operator char*(){ return m_p; }
    char *operator++(){ return ++m_p; }
    char operator [](int i){ return m_s[i]; }
};

int main(){
    Str s("Hi");
    cout << *s << endl;
    ++s;
    cout << s[0] << endl;
    cout << *s << endl;
}
```

The 1st line is (1分)

The 2nd line is (1分)

The 3rd line is (1分)

4-1

4-1

4-2 The output of the code below is:

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

class A{
public:
    A(){cout << "A()" << endl;}
    ~A(){cout << "~A()" << endl;}
};

class B : public A{
public:
    B(){cout << "B()" << endl;}
    ~B(){cout << "~B()" << endl;}
};

int main(){
    A a;
    B b;
}
```

The 1st line is (1分)

The 2nd line is (1分)

The 3rd line is (1分)

The 4th line is (1分)

The 5th line is (1分)

The 6th line is (1分)

4-2

4-3 The output of the code below is:

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

template <typename T>
class FF{
    T a1,a2,a3;
public:
    FF(T b1, T b2, T b3):a1(b1),a2(b2),a3(b3)
    {}
    T Sum() const
    {
        return a1 + a2 + a3;
    }
};

int main()
{
    FF<int> x(2,3,4),y(-2,-3,-4);
    cout << x.Sum() << endl << y.Sum() << endl;
}
```

The 1st line is (1分)

The 2nd line is (1分)

4-3 答案正确 (2分)

4-4 The output of the code below is:

```
#include <iostream>

struct A {
    A() { std::cout << "A" << std::endl; }
    A(const A &a) { std::cout << "B" << std::endl; }
```

```
int main() {
    C c1(7);
    C c2 = 7;
}
```

The 1st line: (1分)

The 2nd line: (1分)

4-5 答案正确 (2分)

4-6 The output of the code below is:

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

class A{
public:
    void F(int){ cout << "A::F(int)" << endl; }
    void F(double){ cout << "A::F(double)" << endl; }
    void F2(int){ cout << "A::F2(int)" << endl; }
};

class B : public A{
public:
    void F(double){ cout << "B::F(double)" << endl; }
};

int main(){
    B b;
    b.F(2.0);
    b.F(2);
    b.F2(2);
}
```

The 1st line is (1分)

The 2nd line is (1分)

The 3rd line is (1分)

4-4

4-5 The output of the code below is:

```
#include <iostream>

class C {
public:
    explicit C(int) {
        std::cout << "i" << std::endl;
    };
    C(double) {
        std::cout << "d" << std::endl;
    };
};

int main() {
```

The 3rd line is (1分)

4-6

4-7 The output of the code below is:

```
#include <iostream>

struct A {
    virtual void foo(int a = 1) {
        std::cout << "A" << '\n' << a;
    }
};

struct B : A {
    virtual void foo(int a = 2) {
        std::cout << "B" << '\n' << a;
    }
};

int main () {
    A *a = new B;
    a->foo();
}
```

The 1st line is (1分)

The 2nd line is (1分)

4-7

4-8 The output of the code below is:

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

template<typename T>
T func(T x,double y){
    return x+y;
}

int s[10];
public:
int operator[](int i)const{
    cout << "operator[](int)const" << endl;return s[i];
}
int &operator[](int i){
    cout << "operator[](int)" << endl;return s[i];
}
};

int main(){
    A a1;
    const A &a2 = a1;
    a1[0] = a2[1];
}
```

The 1st line is (1分)

The 2nd line is (1分)

4-10 答案正确 (2分)

4-11 The output of the code below is:

```
#include <iostream>
using namespace std;
class A{
public:
    static void f(double){
        cout << "f(double)" << endl;
    }
    void f(int){
        cout << "f(int)" << endl;
    }
};

int main(){
    const A a;
    a.f(3);
}
```

```
return x+y;
}

int main(){
    cout << func(2.7,3) << endl;
    cout << func(3,2.7) << endl;
}
```

The 1st line is (2分)

The 2nd line is (1分)

4-8 答案正确 (3分)

4-9 The output of the code below is:

```
#include <iostream>

template<class T> void f(T &i) { std::cout << 1; }

template<> void f(const int &i) { std::cout << 2; }

int main() {
    int i = 24;
    f(i);
}
```

The output is (1分)

4-9 答案正确 (1分)

4-10 The output of the code below is:

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

class A{
    int s[10];
}
```

```
a.f(3);
}
```

The output is (1分)

4-11

4-12 The output of the code below is:

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

class A{
public:
    virtual ~A(){}
};

class B : public A();

int main()
{
    A a;
    B b;

    A *ap = &a;
    if (dynamic_cast<B *>(ap))
        cout << "OK1" << endl;
    else
        cout << "FAIL" << endl;
    if (static_cast<B *>(ap))
        cout << "OK2" << endl;
    else
        cout << "FAIL" << endl;

    ap = &b;
    if (dynamic_cast<B *>(ap))
        cout << "OK3" << endl;
    else
        cout << "FAIL" << endl;
    if (static_cast<B *>(ap))
```

```
    cout << "FAIL" << endl;
    if (static_cast<B *>(ap))
        cout << "OK4" << endl;
    else
        cout << "FAIL" << endl;
}
```

The 1st line is (1分)

The 2nd line is (1分)

The 3rd line is (1分)

The 4th line is (1分)

4-12

4-13 The output of the code below is:

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

class A{
    static int m;
    int n;
public:
    A(int m,int n){
        this->m = m;
        this->n = n;
    }
    void print(){
        cout << m << "---" << n << endl;
    }
};

int A::m;

int main(){
    A a(1,2);
    A b(3,4);
```

```
int A::m;

int main(){
    A a(3,4);
    A a2(5,6);
    a1.print();
    a2.print();
}
```

The 1st line is (2分)

The 2nd line is (1分)

4-13

4-14 The output of the code below is:

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

class A{
public:
    A(){ cout << "A()" << endl;}
    A(const A&){ cout << "A(const A&)" << endl;}
    A &operator=(const A&){
        cout << "operator=(const A&)" << endl;
        return *this;
    }
};

int main()
{
    A a1,a2;
    a2 = a1;
    A a3 = a2;
}
```

The 1st line is (1分)

The 1st line is (1分)

The 2nd line is (1分)

The 3rd line is (1分)

The 4th line is (1分)

4-14

4-15 The output of the code below is:

```
#include <iostream>

struct Base
{
    virtual ~Base()
    {
        std::cout << "Destructing Base" << std::endl;
    }
    virtual void f()
    {
        std::cout << "I'm in Base" << std::endl;
    }
};

struct Derived : public Base
{
    ~Derived()
    {
        std::cout << "Destructing Derived" << std::endl;
    }
    void f()
    {
        std::cout << "I'm in Derived" << std::endl;
    }
};
```

```
std::cout << "Destructing Base" << std::endl;
}
virtual void f()
{
    std::cout << "I'm in Base" << std::endl;
}
};

struct Derived : public Base
{
    ~Derived()
    {
        std::cout << "Destructing Derived" << std::endl;
    }
    void f()
    {
        std::cout << "I'm in Derived" << std::endl;
    }
};

int main()
{
    Base *p = new Derived();
    (*p).f();
    p->f();
    delete p;
}
```

The 1st line: (1分)

The 2nd line: (1分)

The 3rd line: (1分)

The 4th line: (1分)

4-15

公告

这是期末考试的第一部分，在这部分答题时，不能开启考试监考客户端以外的任何软件，包括但不限于浏览器、QQ、微信、支付宝、编程软件等

A 填空题 15

程序填空题 2

5-1

```

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

class Node{
    friend class LinkList;
    Node *m_pNext;
public:
    Node(){ m_pNext = NULL; }
    virtual ~Node() {}
    void AppendNode(Node &n){n.m_pNext = m_pNext; [ ] (1分); }
    virtual void Print()const = 0;
};
class IntNode : public Node{
    int m_i;
public:
    IntNode(int i){m_i = i;}
    virtual void Print()const{cout << m_i << endl;}
};
class StrNode : public Node
{
    char *m_s;
public:
    StrNode(char *s){ m_s = new char[ ] (1分); strcpy(m_s,s);}
    ~StrNode(){ [ ] (1分);}
    virtual void Print()const{cout << m_s << endl;}
};

~StrNode(){ [ ] (1分);}
virtual void Print()const{cout << m_s << endl;}
};
class LinkList
{
    Node *m_pHead;
    Node *m_pEnd;
public:
    LinkList(Node &n){ m_pHead = m_pEnd = &n;}
    ~LinkList(){
        Node *p = m_pHead;
        while (p){ [ ] (1分); delete p; [ ] (1分); }
    }
    void AppendNode(Node &n){
        m_pEnd->AppendNode(n);
        m_pEnd = [ ] (1分);
    }
    void PrintList() [ ] (1分) {
        Node *p = m_pHead;
        while (p){p->Print(); [ ] (1分); }
    }
};

int main()
{
    char word[80];
    cin >> word;
    LinkList llist( [ ] (1分) StrNode(word));
    int i;
    cin >> i;
    llist.AppendNode( [ ] (1分) IntNode(i));
    llist.PrintList();
}

```


5-1

5-2 要求实现一个表示分数的Fraction类中的部分函数，注意，计算结果不要求约分。

```
class Fraction{
    double numerator;
    double denominator;
public:
    Fraction (double numerator=0.0, double denominator=0.0){
        (1分) ;
        (1分) ;
    }
    Fraction ( (1分) f1){
        numerator = f1. numerator;
        (1分) = f1. denominator;
    }
    Fraction (1分) +(const Fraction &f1);
    (1分) operator- ( (1分) );
    return Fraction((numerator*f1.denominator - numerator*f2.denominator) (1分) ), (denominator*f1.denominator));
    (1分) void print(Fraction f);
};
Fraction Fraction::operator+(const Fraction &f1){
    return (1分) ;
}
void print(Fraction f)
{
    cout << "numerator " << f.numerator << endl;
    cout << "denominator " << f.denominator << endl;
}
```

5-2

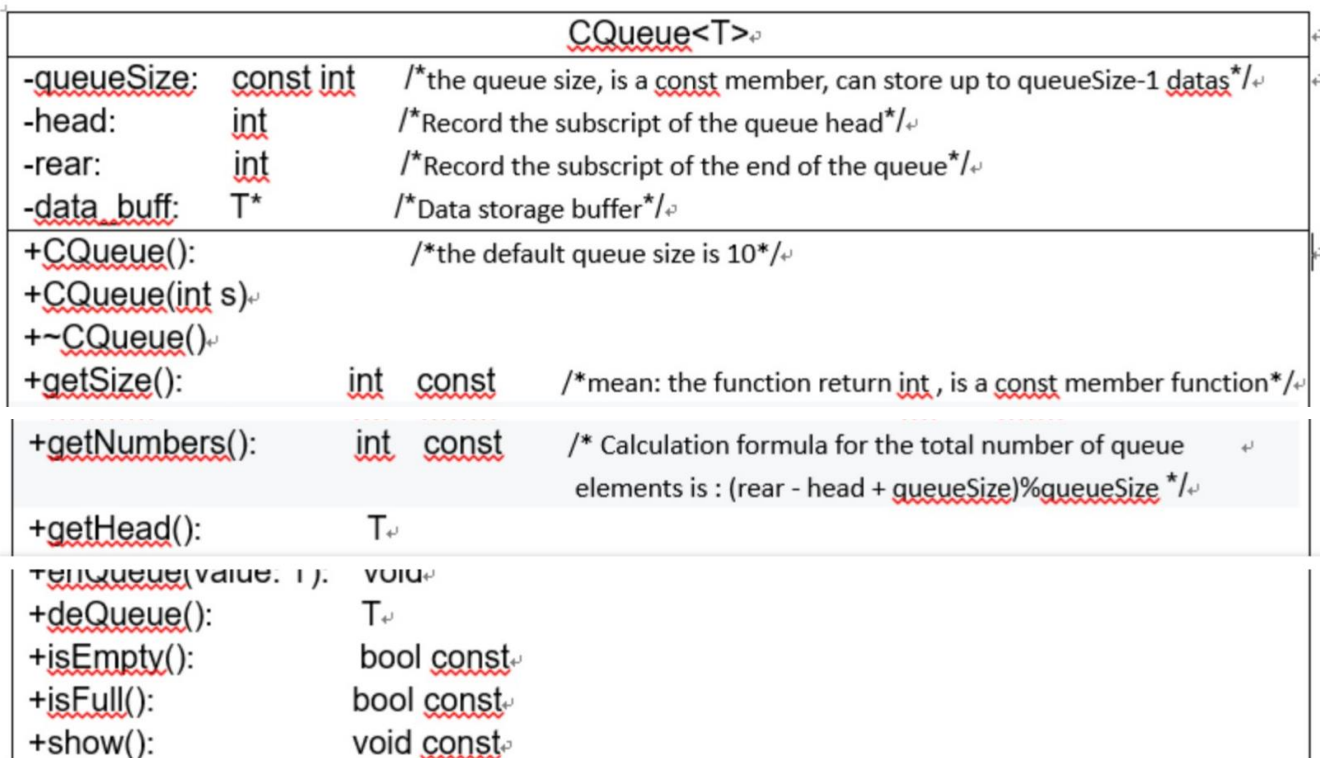
2019-2020学年春夏学期夏末考试第二场 (UD)

[返回](#)

8-1 Queue (35分)

Design a generic circular queue class. using C++ standard exception class: overflow_error and underflow_error in the design.

the picture below shows the UML class diagram for the generic queue class.



The test code is in test.cpp, The output is (There is a space at the end of the line): now the queue is full! 0 1 2 3 4 5 6 7 8
0 5 6 7 8

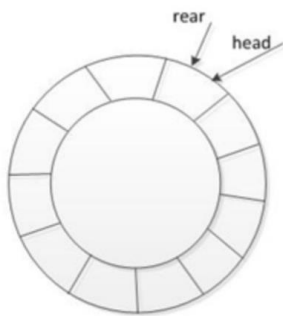


Figure 1 rear==head, The queue is empty.

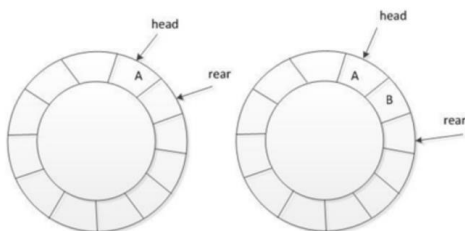


Figure 2 Add elements 'A' and 'B' to the queue in turn.

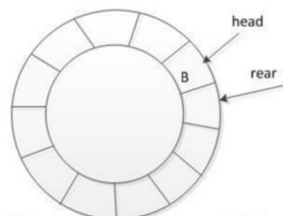


Figure 3 Put the element 'A' of the head out of the queue.

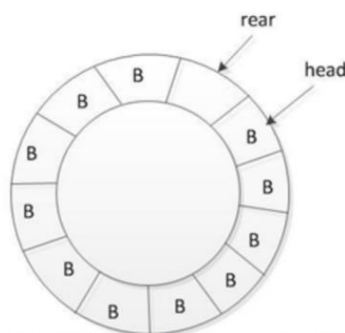


Figure 4 The queue is full when $(rear+1) \% QueueSize == front$.

```

/*test.cpp*/
#include <iostream>
#include <stdexcept>
using namespace std;
#include "CQueue.h"
int main()
{
    try {
        CQueue<double> rq;
        for (int i = 0; i < rq.getSize()-1; i++)
            rq.enqueue(i);
        if (rq.isFull()) printf("now the queue is full! ");
        if (!rq.isEmpty()) rq.show();
        cout << rq.getHead() << " ";
        for (int i = 0; i < 5; i++) // dequeuing 5 elements
            rq.dequeue();
        rq.show();
    }
    catch (overflow_error& r)
    {
        cout << r.what();
    }
    catch (underflow_error& r)
    {
        cout << r.what();
    }
    return 0;
}

```