东 南 大 学 考 试 卷（A 卷）

课程名称 计算机科学基础 II 考试学期 12-13-3 得分 73

适用专业 信息工程 考试形式 闭卷 考试时间长度 120 分钟

题目	一	二	三	四	总分
得分	14	4	21	34	73
批阅人	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

自觉遵守考场纪律 如考试作弊 此答卷无效

姓名 [illegible]

学号 04212137

线 封 密

一、单选题：（请将答案填写在下面表格中。每空 2 分，共 20 分）

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
C	B	B	B	D	B	D	D	A	C

1. 下列有关类和对象的说法中，正确的是____(1)____。
   A. 类和对象没有区别
   B. 要为类和对象分配存储空间
   C. 对象是类的实例，为对象分配存储空间而不为类分配存储空间
   D. 类是对象的实例，为类分配存储空间而不为对象分配存储空间
2. 关于构造函数，以下正确的说法是____(2)____。
   A. 定义类的成员时，必须定义构造函数，因为创建对象时，系统必定要调用构造函数
   B. 对象一经说明，首先调用构造函数，如果类中没有定义构造函数，系统会自动产生一个不做任何操作的缺省构造函数
   C. 无参构造函数和参数为缺省值的构造函数符合重载规则，因此一个类中可以含有这两种构造函数
   D. 构造函数没有返回值，因为系统隐含指定它的返回值类型为 void
3. 对线性表，在下列哪种情况下应当采用链表表示？____(3)____。
   A. 经常需要随机地存取元素　　B. 经常需要进行插入和删除操作
   C. 表中元素需要占据一片连续的存储空间　　D. 表中元素的个数不变
4. 执行以下程序时，调用构造函数（简称"构造"）和复制构造函数（简称"复制"）的次数分别为____(4)____。

```
class Test{
    int x,y;
public:
    Test(int a,int b){  x=a;  y=b;  cout<<"调用构造函数!\n";  }
    Test(Test &t){  x=t.x;  y=t.y;  cout<<"调用复制构造函数！\n"; }
    void show(){  cout<<"x="<<x<<'\t'<<"y="<<y<<'\n';  }
};
int main(){
    Test t1(1,1), t2(2,2);
    t2=t1;
    Test t3(t1);
}
```

   A. 1 次构造，1 次复制　　B. 1 次构造，2 次复制
   C. 2 次构造，1 次复制　　D. 2 次构造，2 次复制
5. 复制构造函数有深复制和浅复制之分，二者的区别在于____(5)____。
   A. 深复制能用"="运算符进行对象的复制，而浅复制不能
   B. 深复制能对成员数据进行初始化，而浅复制不能
   C. 浅复制不能复制指针型的成员数据，而深复制可以
   D. 浅复制使对象共享动态分配的资源，而深复制为对象分配独自拥有的资源
6. 编译时的多态性可以通过使用____(6)____获得。
   A. 虚函数和指针　B. 重载函数和析构函数　C. 虚函数和对象　D. 虚基类和引用
7. 下列描述中哪个是正确的。____(7)____。
   A. 私有派生的子类无法访问父类的成员
   B. 类 A 的私有派生子类的派生类 C 无法初始化其祖先类 A 对象的属性，因为类 A 的成员对类 C 是不可访问的
   C. 私有派生类不能作为基类派生子类
   D. 私有派生类的所有子孙类将无法继续继承该类的成员
8. 若某二叉树的前序遍历访问顺序是 abdgcefh，中序遍历访问顺序是 dgbaechf，则其后序遍历的节点访问顺序是____(8)____。
   A. bdgcefha　B. gdbecfha　C. bdgaechf　D. gdbehfca
9. 已知在函数 set 中，有语句 this->x=5；与语句 x=5；的效果完全相同。根据这一结论，以下叙述中不正确的是____(9)____。
   A. x 是某个类的数据成员，set 是该类的友元函数
   B. x 是某个类的数据成员，set 是该类的成员函数
   C. set 不是该类的静态成员函数
   D. x 不是该类的常成员数据
10. 设栈 S 的初始状态为空，元素 E1、E2、E3、E4、E5 和 E6 依次进入栈 S，若出栈的顺序为 E2、E4、E3、E6、E5 和 E1，则栈 S 的容量至少应该是____(10)____。
   A. 6　B. 4　C. 3　D. 2

二、填空题（每空 2 分，共 10 分）

1. 请写出下面程序段，1)、2)、3) 三行的结果。

```
fstream file("ff.dat", ios::out |ios::in| ios::binary);
char ch[ ]="How to Program";
for(int i=0;i<sizeof(ch);i++)    file1.write(&ch[i],sizeof(char));
int pos=file1.tellp();
cout<<"当前指针位置是: "<<pos<<endl;    //1)
char ch1[10];
file1.seekg(-8,ios::end);
pos=file1.tellp();
cout<<"当前指针位置是: "<<pos<<endl;    //2)
for(i=0;i<pos;i++){
                    file1.read(&ch1[i],sizeof(char));
                    cout<<ch1[i];      //3)
}
```

1) 当前指针位置是 60 -1

2) 当前指针位置是 32 -1

3) How to P X -2

2. 设一组初始关键字序列为（49,38,65,97,76,13,27,49），对其按升序进行冒泡排序，请写出排过三趟后的结果 (13,27,38,49,65,97,76,49) X -2。

3. 设一组初始关键字序列为（8,6,7,9,4,5,2），对其按升序进行插入排序，请写出排过二趟后的结果 {6,7,8,9,4,5,2} √ -1

二. 读程序写结果（共 30 分）

1.阅读以下程序，写出运行结果（每空 1 分，共 5 分）

```
#include<iostream>
#include<string>
using namespace std;
class worker{
    int num;
    float wage;
    string level;
public:
    worker(int=0,float=0);
    worker(worker &);
    void List();
    void Set(int=2000);
    void Up();
};
worker::worker(worker &a){
    num=a.num;
    wage=a.wage;
    level=a.level;
}
worker::worker(int ID,float WG){
    num=ID;
    wage=WG;
    if(wage>=5000) level="ALVL";
    else if(wage>=4000) level="BLVL";
    else if(wage>=3000) level="CLVL";
    else level="NULL";
}
void worker::List(){
    cout<<"num:"<<num
        <<"\twage:"<<wage
        <<"\tlevel:"<<level<<endl;
}
void worker::Set(int WG){
    wage=WG;
    if(wage>=5000) level="ALVL";
    else if(wage>=4000) level="BLVL";
    else if(wage>=3000) level="CLVL";
    else level="NULL";
}
void worker::Up(){
    wage+=1000;
    if(wage>=5000) level="ALVL";
    else if(wage>=4000) level="BLVL";
    else if(wage>=3000) level="CLVL";
    else level="NULL";
}
int main(){
    worker Joe(1,2000);
    Joe.List();
    Joe.Set(3500);
    Joe.List();
```

1. 输出结果：

num:1 wage:2000 level: NULL √

num:0 wage:3500 level: CLVL X

num:0 wage:2000 level: NULL √

num:1 wage:2000 level: NULL X

num:1 wage:3000 level: CLVL X

-3

```
    worker Sarah;
    Sarah.Set();
    Sarah.List();
    worker Aska=Joe;
    Aska.List();
    Aska.Up();
    Aska.List();
    return 0;
}
```

2. 阅读以下程序，写出运行结果（每空 1 分，共 5 分）

```
#include<iostream>
using namespace std;
class RMB{
    int yuan,jiao,fen;
public:
    RMB(int y=0,int j=0,int f=0);
    operator float();
    RMB &operator +(RMB &);
    void modify(int a,int b,int c);
    friend ostream& operator<<(ostream&,RMB&);
};
RMB::RMB(int y,int j,int f){
    yuan=y;
    jiao=j;
    fen=f;
}
void RMB::modify(int a,int b,int c){
    yuan=a;
    jiao=b;
    fen=c;
}
RMB::operator float(){
    return float(yuan+0.1*jiao+0.01*fen);
}
RMB & RMB::operator +(RMB &rmb){
    RMB sum(rmb);
    rmb.fen=(rmb.fen+fen)%10;
    rmb.jiao=(rmb.jiao+jiao+(rmb.fen+fen)/10)%10;
    rmb.yuan=rmb.yuan+yuan+(rmb.jiao+jiao+(rmb.fen+fen)/10)/10;
```

2. 输出结果：

转换前：7元8角9分
转换后：7.89
9元8角3分
9.83
17元4角2分


```
    return rmb;
}
ostream& operator<<(ostream& out,RMB& r){
    out<<r.yuan<<"元"<<r.jiao<<"角"<<r.fen<<"分"<<endl;
    return out;
}
int main(){
    RMB r(7,8,9),m;
    cout<<"转换前："<<r;
    float z=float(r);
    cout<<"转换后："<<z<<"元"<<endl;
    m.modify(9,5,3);
    z=float(m);
    cout<<m;
    cout<<z<<endl;
    cout<<r+m;
    return 0;
}
```

3. 阅读以下程序，写出运行结果（每空 2 分，共 14 分）

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
class point
{
private:
    float x;
protected:
    float y;

public:
    void show(){cout<<"point:"<<x<<","<<y<<endl;}
    point(float a,float b){x=a;y=b;}
    void setp(float a=0,float b=0){x=a;y=b;}
    float getx(){return x;}
    float gety(){return y;}
    virtual float area(){cout<<"point area = 0\n"; return 0;}
    ~point();
};
```

3. 输出结果：

point : 2,9
point area=0
point : 1,1
another point : 2,2
rect area=1
rect destructor!
point destructor!
point destructor!

```
point::~point(){  cout<<"point destructor!\n";    }

class rect :public point{
private: float x,y;
public:
    rect(float a,float b,float c,float d):point(a,b),x(c),y(d){}
    void setp(float a=1,float b=1,float c=2,float d=2){point::setp(a,b);x=c;y=d;}
    float geth(){return fabs(y-gety());}
    float area(){
        float a=fabs(x-getx())*geth();
        cout<<"rect area="<<a<<endl;
        return a;
    }
    void show();
    ~rect();
};
void rect::show(){
    point::show();
    cout<<"another point:"<<x<<","<<y<<"rect area"<<area()<<endl;
}
rect::~rect(){    cout<<"rect destructor!\n";    }
int main(){
    point pobj(2,9),*p;
    p=&pobj;
    p->show();
    p->area();
    rect robj(7,5,4,4);
    p=&robj;
    p->setp();
    p->show();
    p->area();
    return 0;
}
```

4. 阅读以下程序，写出运行结果（每空 2 分，共 6 分）

```
#include <iostream>
using namespace std;
double   reciprocal(int x, int y);//求 x/y 的倒数
void main( ){
    try
```



```
    {   cout<<"Reciprocal of 4/3: "<< reciprocal(4,3)<<endl;
        cout<<"Reciprocal of 0/6: "<< reciprocal(0,6)<<endl;
        cout<<"Reciprocal of 2/10: "<< reciprocal(2,10)<<endl;
    }
    catch(int x)
    {   cout<<"numerator is "<<x<<endl; }
    cout<<"The end?\n";
}

double   reciprocal(int x,int y){
    if(x==0)
        throw x;
    return y*1.0/x;
}
```

4. 输出结果：

Reciprocal of 4/3: 0.75
Reciprocal of 0/6: numerator is 0
The end?

四. 完善程序题（每空 2 分，共 40 分）

1. 已知数组 buffer 中共有 m+n 个元素，其中前 m 个元素已是升序排列，后 n 个元素是乱序排列，模板函数 sort 实现对 buffer 的升序排序，其使用的方法是对后 n 个元素做冒泡排序，每次冒出的那个元素调用模板函数 insert 用对半查找方法插入到前面已按升序排序元素序列中。请完善程序。

```
template<typename T2>void sort(T buffer [], int m, int n){
    for (int ____i=m_____; i < m + n; i++) {
        for (int j = m + n - 1; j > i; j--) {
            if (buffer[j] < buffer[j - 1]) {
                T temp = buffer[j];

                buffer [j] = ___buffer[j-1]___;
                buffer [j - 1] = temp;
            }
        }
        insert (buffer, i);
    }
}

template <typename T2> void insert(T2 list [], int k)
{
    T2 temp = list[k];
    int low = 0, high = ____k-1_____;
    while (low <= high) {
```

```
        int ____mid = (low+high)/2____
        if (temp < list[mid])
            high = mid - 1;
        else
            low = mid + 1;
    }
    for (int j = k - 1; j >= low; j--)
        list[j + 1] = list[j];

    ____list[low]____ = temp;
}
```

2.以下为循环字节队列类 Queue 定义，其数据成员 rear 和 front 分别是最后一次进队和最后一次出队元素所在位置索引，elements 为指向存储队列的动态数组的指针，size 为该动态数组元素个数；其成员函数 isEmpty 和 isFull 分别判断该队列是否空和满，length 返回队列目前容纳的元素个数，push 和 pop 分别实现元素进队和出队功能，进队和出队元素用 data 表示，而两者的返回值在操作成功时返回 true，否则分别在队已满和空时返回 false。队列实际容纳元素个数在构造队列对象时给出，而申请的动态数组元素个数要比前者多一个。请完善程序。

```
class Queue{
    int rear, front;
    char * elements;
    int size;
public:
    Queue(int s = 31){
        size = s + 1;
        elements = ____new char[size]____;
        rear = front = 0;
    }
    ~Queue() {    delete [ ] elements;    }

    bool isEmpty() {    return ____rear == front____;    }

    bool isFull() { return (rear + 1) % size == front; }

    int length() { return ____(rear-front+size)%size____

    bool push(const char data){
        if (isFull ())
            return false;
        else {
            rear = (rear + 1) % size;
```



```
            elements[rear] = data;
            return true;
        }
    }

    bool pop(char & data){
        if (isEmpty ())    ____return false____;
        else {
            front = (front + 1) % size;
            data = elements[front];
            return true;
        }
    }

    bool getFront(char & data){
        if (isEmpty ())
            return false;
        else {
            data = elements[(front + 1) % size];
            return true;
        }
    }
    void empty() { front = rear = 0; }
};
```

3. 以下程序实现有头链表类 List，辅助节点类 Node 完成存储整数的功能。类 List 包含头节点指针 head 和尾节点指针 tail，成员函数 empty 清空链表，insert 实现前向（即表头）插入，reverse 实现链表节点链接顺序翻转，其使用的算法是每次从原表取出尾部节点并在新表做后向插入（即表尾）插入。请完善程序。

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

class ____List____;
class Node{
    int data;
    Node * next;
public:
    Node(int info = 0){
        data = info;
        next = NULL;
    }
    Node * getNext(){    return    next;    }
```

```
    friend List;
    friend ostream & operator<<(ostream & sink, Node & one){
        sink<<one.data;
        ____return sink____;
    }
};

class List{
    Node * head;
    Node * tail;
public:
    List() { tail = head = new Node; }
    ~List(){
        empty ();
        ____delete head____;
    }

    void empty(){
        Node * p = head->next;
        while (p) {
            ____head = p____;
            p = p->next;
            delete q;
        }
    }

    void insert(Node * item){
        item->next = head->next;
        ____head->next____ = item;
        if (head = = tail)
            ____tail=item____;
    }
    friend ostream & operator<<(ostream & sink, List & one);
    void reverse();
};
```



```
ostream & operator<<(ostream & sink, List & one){
    Node * temp = one.head->getNext ();
    int count = 0;
    while (temp) {
        count++;
        sink<<count<<'\t'<<*temp<<endl;
        temp = ____temp->next____;
    }
    return sink;
}

void List::reverse(){
    Node * h = head->next;
    ____head=tail____;
    head->next = NULL;
    while (h) {
        Node * q = h;
        Node * p = h -> next;
        while (p && p->next) {
            q = p;
            p = p->next;
        }
        Node * item = NULL;
        if (p){
            item = p;
            q->next = NULL;
        }
        else {
            item = q;
            ____h= NULL____;
        }
        ____tail->next____ = item;
        tail = item;
    }
}
```