

1. In C++ , There is a kind of class that it can only be inherited but cannot be initiated, we call that class as a (      ), to define that kind of class, it has to have at least a (      ) 。  
 (A) virtual function      **(B) pure virtual function**      (C) abstract class      (D) friend class
2. Assume that AB is a class, the declaration of copy constructor of that class should be (      ) 。  
 (A) AB&(AB x)      (B) AB(AB x)      **(C) AB(AB &x)**      (D) AB(AB \*x)
3. For the member of a structure, the default access modifier is (      ) 。  
**(A) public;**      (B) private;      (C) protected;      (D) static;
4. Assume AB is a class, to execute statement “AB a(4),b[3],\*p[2];”, how many times to invoke the constructor?(      ) 。  
 (A) 3      **(B) 4**      (C) 6      (D) 9
5. Which of the following is not correct (      )  
 (A) The name of constructor is same as the name of the class  
 (B) class constructor could have default parameters  
**(C) the default return type of a constructor is of type int**  
 (D) the constructor can be overloaded
6. Parameterized stream manipulator ***setfill*** specifies the fill character that is displayed when an output is displayed in a field wider than the number of characters or digits in the output. The effect of ***setfill*** applies:  
 (A) Only to the current value being displayed.  
 (B) Only to outputs displayed in the current statement.  
**(C) Until explicitly set to a different setting.**  
 (D) Until the output buffer is flushed.
7. Assuming the following is the beginning of the constructor definition for class Circle which inherits from class Point,  
 Circle::Circle( double r, int a, int b )  
       : Point( a, b )  
 The second line:  
**(A) Invokes the Point constructor with values a and b.**  
 (B) Causes a compiler error.  
 (C) Is unnecessary because the Point constructor is called automatically.  
 (D) Indicates inheritance.
8. Which of the following assignments would be a compilation error?  
 (A) Assigning the address of a base-class object to a base-class pointer.  
**(B) Assigning the address of a base-class object to a derived-class pointer.**  
 (C) Assigning the address of a derived-class object to a base-class pointer.  
 (D) Assigning the address of a derived-class object to a derived-class pointer.

9. Assume class T is declared as following, which of the following declaration of the function fF() is not correct(            )。

```
class T
{ int i;
  friend void fF(T&,int) ;
};
```

- (A) void fF (T &objT,int k) { objT.i = k+1; }
- (B) void fF (T &objT,int k) { k = objT.i+1; }
- (C) **void T::fF (T &objT,int k) { k += objT.i; }**
- (D) void fF (T &objT,int k) { objT.i += k; }

**Find out errors in the following code, and correct them.**

1) Following is the destructor function prototype in Time class definition:

```
void ~Time( int );                    ~Time(    );
```

2) Following is part of the Time class definition:

```
class Time
{ public:
  private:
    int hour = 0;
    int minute = 0;
    int second = 0;
}; // end class Time
```

```
class Time
{ public:
  private:
    Static int hour ;
    Static int minute;
    Static int second ;
}; // end class Time
```

**Write the output of the following program.**

```
1
#include <iostream>
using namespace std;
class test{
  private:
    int num;
    float fl;
  public:
    test();
    int getint(){ return num;}
    float getfloat(){ return fl;}
    ~test();
```

```

};

test::test(){
    cout<<"Initailizing default"<<endl;
    num = 0;
    fl = 0.0;
}
test::~~test(){
    cout<<"Destructor is active" <<endl;
}

int main(){
    test array[2];
    cout<<array[1].getint()<< " " <<array[1].getfloat()<<endl;
}

```

**Initailizing default**

**Initailizing default**

**0 0**

**Destructor is active**

**Destructor is active**

2.

```

#include <iostream>
using namespace std;
class A {
public:
    A() { cout <<" A::A() called.\n";}
    virtual ~A() { cout<<"A::~~A() called.\n";}
};

class B: public A{
public:
    B(int i){
        cout<<"B::B()called.\n";
        buf = new char[i];}
    virtual ~B() {
        delete [] buf;
        cout<< "B::~~B() called.\n";
    }
private:
    char * buf;
};

void fun(A * a) {

```

```

        delete a;
    }
    void main() {
        A * a = new B(15);
        fun(a);
    }

```

**A::A() called.**

**B::B()called.**

**B::~~B() called.**

**A::~~A() called.**

3.

```

#include<iostream>
using namespace std;
class T {
public:
    T(int x){ a=x; b*=x;};
    static void display(T c){
        cout<<"a="<<c.a<<"\t"<<"b="<<c.b<<endl; }
private:
    int a;
    static int b;
} ;
int T::b=4;
int main(){
    T A(3),B(2);
    T::display(A);
    T::display(B);
}

```

**a=3 b=24**

**a=2 b=24**

4.

```

#include <iostream>
using namespace std;
class B
{   int b;
public:
    B(int i) { b=i; }
    virtual void virfun() { cout<< "B::b: "<<b<<endl; }
};
class D: public B

```

```

{   int d;
public:
    D(int i,int j): B(i) { d=j; }
    void virfun() { B::virfun(); cout<<"D::d: "<<d<<endl; }
};
void fun(B *objp) { objp->virfun(); }
void main()
{ B *pd ;
  pd=new B(3) ;    fun(pd);
  pd=new D(5,7);   fun(pd);
}

```

**B::b: 3**

**B::b: 5**

**D::d: 7**

5.

```

#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

class Employee
{ public:
    Employee(const long k ,const char*   str ){ number = k;strcpy_s(name,20,str); }
    virtual ~Employee(){ name[0] = '\0';}
    const char * getName() const{ return name;}
    const long getNumber() const { return number;}
        virtual double earnings() const=0;
        virtual void print() const { cout <<number<<setw(16)<<name ;}
        Employee * next;

    protected:
        long number;
        char name[20];
};

class Manager : public Employee
{ public:
    Manager(const long k , const char * str, double salary):   Employee(k,str)
    { setMonthlySalary(salary);}
    ~Manager() { }
    void setMonthlySalary(double salary) { monthlySalary = salary;}
}

```

```

        virtual double earnings() const { return monthlySalary;}
        virtual void print() const { Employee::print(); cout<<setw(16)<<"Manager\n";}

private:
    double monthlySalary ;
};

class HourlyWorker : public Employee
{ public:
    HourlyWorker(const long k , const char * str, double w=0.0, int h=0 ): Employee(k,str){
        setWage(w); setHours(h);}
    ~HourlyWorker(){ }
    void setWage(double w) { wage = w ;}
    void setHours(int h)    { hours = h ;}
    virtual double earnings() const { return wage * hours ;}
    virtual void print() const { Employee::print();
        cout<<setw(16)<<"Hours   Worker\n"<< " \t\twageperhour   "<<wage<<"   Hours
"<<hours;
        cout<<"   earned $"<<earnings()<<endl; }
private:
    double wage;
    double hours;
};

class PieceWorker : public Employee
{ public:
    PieceWorker(const long k , const char * str, double wage =0.0, int
quantity=0 ):Employee(k,str){
        setWage(wage); setQuantity(quantity);}
    ~PieceWorker() { }
    void setWage ( double wage ){ wagePerPiece = wage ;}
    void setQuantity ( int q){ quantity = q ;}
    virtual double earnings() const { return wagePerPiece * quantity; }
    virtual void print() const { Employee::print();
        cout<<setw(16) << "Piece Worker\n" ;
        cout<<" \t\twagePerPiece   "<< wagePerPiece<<"   quantity   "<<quantity;
        cout<<"   eared $" << earnings() <<endl;}
private:
    double wagePerPiece;
    int quantity;
};

void main()
{ Employee * employ[3] ;

```

```

int i;
employ[0] = new Manager( 10135, "ZhangSan", 1200 ) ;
employ[1] = new HourlyWorker( 30712, "LiSi", 5, 260 ) ;
employ[2] = new PieceWorker( 20382, "Wangwu", 0.5, 2850 ) ;
cout << setiosflags(ios::fixed|ios::showpoint) << setprecision(2) ;
for( i = 0; i <3; i ++ )
    employ[i] -> print() ;
for( i = 0; i < 3; i ++ )
    cout <<setw(8)<< employ[i]->getName() << " " <<setw(10)<< employ[i] -> earnings()
<< endl ;

}

```

```

10135      ZhangSan      Manager
30712      LiSi    Hours Worker
           wageperhour  5.00  Hours  260.00   earned $1300.00
20382      Wangwu    Piece Worker
           wagePerPiece  0.50  quantity  2850   eared $1425.00
ZhangSan   1200.00
LiSi       1300.00
Wangwu     1425.00

```

### fill in the blanks according to the output

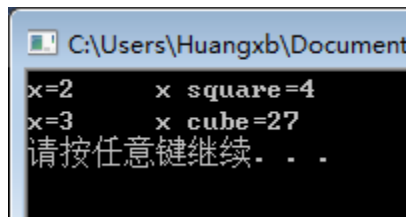
1 . The following program defined a complex class stands for complex number. It has real part real and imaginary part img. It overloaded complex number addition by using friend function. Please finish it.

```

class complex{
    float    [1],    [2];    real, img
public:
    complex (float r=0,float i=0)
    {    real=r;    img=i;    }
    [3] [4] operator+(complex c1,complex c2);    friend complex
};
[5] operator+(complex c1,complex c2)    complex
{    complex temp;
    [6]    temp.real=c1.real+c2.real;
    [7]    temp.img=c1.img+c2.img;
    return temp;
}

```

2. The following code has the screen output below, please fill in the blanks.



```
#include <iostream>
#include <math>
using namespace std;
class Power
{ public:
    Power(int i){x=i;}
    ____ (8) virtual ____ void display(){ cout<<"x="<<x;} // definition of virtual function
protected:
    int x;
};
class Square: public Power
{ public:
    Square(int n): ____ (9) Power(n) ____ { } //Constructor of Square
    void display()
    { ____ (10) Power::display() ____
      cout<<"\tx square="<<x*x<<endl;}
};
class Cube:public Power
{ public:
    Cube(int n):Power(n){ }
    void display()
    { ____ (10) Power::display() ____;
      cout<<"\tx cube="<<x*x*x<<endl;}
};
void fun(Power &p)    { p.display();}
void main()
{ Square squ(2);
  Cube cub(3);
  fun(squ);
  fun(cub);
}
```

## Write program

1. Use function template to calculate average of arrays of different data type. Function main ()



is as following:

```
#include <iostream>
using namespace std;
void main()
{   int a[]={ 1,2,3,4,5,6,7,8,9,10 };
    double b[]={ 1.1,2.2,3.3,4.4,5.5,6.6,7.7,8.8,9.9,10.0 };
    cout<< "Average of array a:" <<average(a,10)<<endl;
    cout<< "Average of array b:" <<average(b,10)<<endl;
}
```

Program calculates average of an array of type integer and of type double. complete a function template to accomplish such function(功能) (6 points )

```
template<typename T>
T average(T*arr, int num)
{
    T sum = arr[0];
    for (int i = 0; i < num; i++)
    {
        sum += arr[i];
    }
    T result = sum / num;
    return result;
}
```

2.

(1) Rewrite class Student, overload operator >> and << to substitute(代替) function input() and output() respectively. (8 points)

(2) Add a constructor for class Student to initiate data members ; (3 points)

(3) Modify function main() to get correct result. (3 points)

```
#include <iostream>
using namespace std;
class Student
{
    char name[20];
    unsigned id;
    double score;
public:
```

```

        void input()
        {
            cout<<"name? ";
            cin>>name;
            cout<<"id? ";
            cin>>id;
            cout<<"score? ";
            cin>>score;
        }
        void output()
        {
            cout<<"name: "<<name<<"\tid: "<<id<<"\tscore: "<<score<<endl;
        }
    };
    int main()
    {
        Student s;
        s.input();
        s.output();
    }

```

```

#include <iostream>

#include "string.h"

using namespace std;

class Student
{
    char name[20];

    unsigned id;

    double score;

public:
    Student(){}

    Student(char*n, unsigned i, double s)
    {
        strcpy_s(name, n);

        id = i;

        score = s;
    }

    friend ostream& operator<<(ostream& output, const Student&stu)
    {

```

```

        output << "name: " << stu.name << "\tid: " << stu.id << "\tscore: " << stu.score << endl;

        return output;
    }

    friend istream& operator>>(istream& input, Student &stu)
    {
        cout << "name? ";

        input >> stu.name;

        cout << "id? ";

        input >> stu.id;

        cout << "score? ";

        input >> stu.score;

        return input;
    }
};

int main()
{
    Student stu;

    cin >> stu;

    cout << stu;

}

```