

## Object oriented programming

Task1:	Task2:
Task3:	Task4:
Task5:	Task6:
Task7:	

Task1	Task2	Task3	Task4	Task5	Task6	Task7	Sum

## **Task 1. What is the output for the following program?**

```
#include <iostream>
using namespace std;
class Student;
class Person {
public:
    Person(){cout << "constructor Person" << endl;}
    ~Person(){cout << "destructor Person" << endl;}
};
const Student& returnPerson(const Student& p){return p;}
class Student : public Person {
public:
    Student(){cout << "constructor Student" << endl;}
    ~Student(){cout << "destructor Student" << endl;}
};
Student returnStudent(Student s){ return s;}
class PhDStudent : public Student {
public:
    PhDStudent(){cout << "constructor PhDStudent" << endl;}
    ~PhDStudent(){cout << "destructor PhDStudent" << endl;}
};
Student returnPhDStudent(Student s){ return s; }

int main(int argc, char* argv[])
{
    PhDStudent laza;
    returnPhDStudent(returnStudent(returnPerson(laza)));
}
```

## **Task 2. What is the output for the following program?**

```
#include <iostream>
#include <complex>
using namespace std;
class Base{
public:
    virtual void f( int );
    virtual void f( double );
    virtual void g( int i = 10 );
};
void Base::f( int ){ cout << "Base::f(int)" << endl;}
void Base::f( double ){ cout << "Base::f(double)" << endl;}
void Base::g( int i ){ cout << i << endl;}

class Derived: public Base{
public:
    void f( complex<double> );
    void g( int i = 20 );
};
void Derived::f( complex<double> ){ cout << "Derived::f(complex)" << endl;}
void Derived::g( int i ){ cout << "Derived::g() " << i << endl;}

void main(){
    Base    b;   Derived d;   Base*   pb = new Derived;
    b.f(1.0);
    d.f(1.0);
    pb->f(1.0);
    b.g();
    d.g();
    pb->g();
    delete pb;
}
```

### **Task 3. What is the output for the following program?**

```
#include <iostream>
using namespace std;
class Base{
    int x;
    virtual int f( int );
public:
    Base( int a ) : x(a) {}
    double f( double );
};
int Base::f( int a ){ return a+x;}
double Base::f( double a ){ return a*x + f(5);}

class Derived : public Base{
    int x;
    int f( int );
public:
    Derived ( int a, int b ) : Base(a), x(b) {};
    double f( double );
};
int Derived::f( int a ){ return a-x;}
double Derived::f( double a ){ return a/x+f(5); f(1);}

void main()
{
    Base b(2);
    Derived d(2,4);
    Base* pb = new Derived(2,4);
    cout << b.f( 1.2 ) << endl;
    cout << d.f( 1.2 ) << endl;
    cout << pb->f( 1.2 ) << endl;
}
```

#### **Task 4. What is the output for the following program?**

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

class Base{
public:
    int x;
    Base(){x = 5;}
    virtual int a();
    int b();
    void operator() (Base& x);
};

int Base::a(){cout << "Base A\n";this->x;return b();}
int Base::b(){cout << "Base B\n";return this->x;}
void Base::operator() (Base& x){
    this->x = x.a() * x.b();
    cout << this->x << endl;
}

class Derived: public Base{
public:
    int a();
    int b();
};

int Derived::a(){    cout << "Derived A\n";return b()*100;}
int Derived::b(){    cout << "Derived B\n";return this->x * 10;}

int main(){
    Base obj;
    Base *d = new Derived();
    obj.a(); d->a();    obj.b();
    d->b(); obj(*d); (*d)(obj);
    return 0;
}
```

### **Task 5. What is the output for the following program?**

```
#include <iostream>
using namespace std;
static int b;
class X {
    int *pi;
public:
    X(){b=0;};
    X(int i) : pi(new int(i)) {b++;}
    X(const X &x) : pi(new int(*x.pi)){b*=2;}
    X& operator= (const X&);
    ~X() {delete pi;}
};
X& X::operator= (const X& x) {
    if (this != &x) {
        delete pi;
        pi = new int(*x.pi);
    };
    return *this;
}

X funF(X x){X Xnew(5); x=Xnew; return x;}

void g() {
    X xa=3, xb=1;
    X xc = xa;
    xa = funF(xb);
    xc = xa;
}

int main(int argc, char* argv[]){
    X d();
    g();
    cout << b << endl;
    return 0;
}
```

## **Task 6. What is the output for the following program?**

```
#include <iostream>
using namespace std;
int n = 3, h0 = 256, equalCoo = 0;
static int mX,mY,mH;
class DrawA; class DrawB; class DrawC; class DrawD;
class Draw{
public:
    Draw(bool ord, int sign);
    ~Draw(){};
    void draw(int i, Draw& first, Draw& second);
    virtual void mainDraw(int level){};
protected:
    void equal(){if(mX == mY) {equalCoo++;cout << mX << endl;}}
    bool mOrd;
    int mSign;
    int mLevel;
};

Draw::Draw(bool ord, int sign) : mOrd(ord), mSign(sign) {};
void Draw::draw(int i, Draw &first, Draw &second){
    if (i > 0){
        if(mOrd){
            first.mainDraw(i-1); mX = mX - mSign * mH;equal();
            draw(i-1, first, second); mY = mY - mSign * mH;equal();
            draw(i-1, first, second); mX = mX + mSign * mH;equal();
            second.mainDraw(i-1);
        }else{
            first.mainDraw(i-1); mY = mY - mSign * mH;equal();
            draw(i-1, first, second); mX = mX - mSign * mH;equal();
            draw(i-1, first, second); mY = mY + mSign * mH;equal();
            second.mainDraw(i-1);
        }
    }
}

class DrawA : public Draw{
public:
    DrawA(bool ord, int sign) : Draw(ord, sign){}
    void mainDraw(int level);
};

class DrawB : public Draw{
public:
    DrawB(bool ord, int sign) : Draw(ord, sign){}
    void mainDraw(int level);
};

class DrawC : public Draw{
public:
    DrawC(bool ord, int sign) : Draw(ord, sign){}
    void mainDraw(int level);
};

class DrawD : public Draw{
public:
    DrawD(bool ord, int sign) : Draw(ord, sign){}
    void mainDraw(int level);
};

void DrawA::mainDraw(int level){DrawD d(false,1); DrawB b(false, -1); draw(level,d,b);}
void DrawB::mainDraw(int level){DrawC c(true, -1);DrawA a(true, 1); draw(level,c,a);}
void DrawC::mainDraw(int level){DrawB b(false,-1); DrawD d(false, 1); draw(level,b,d);}
void DrawD::mainDraw(int level){DrawA a(true,1); DrawC c(true,-1); draw(level,a,c);}

int main(int argc, char* argv[]){
    int step = 0, hlen = h0, x0 = hlen / 2, y0 = x0, i = 0;
    while(true){
        i++;
        hlen = hlen / 2;
        x0 = x0 + (hlen / 2);
        y0 = y0 + (hlen / 2);
        mX = x0; mY = y0; mH = hlen;
        DrawA A(true, 1);
        A.mainDraw(i);
        if ( i == n ){
            cout << equalCoo << endl;
            return 1;
        }
    }
    cout << equalCoo << endl;
    return 0;
}
```

## **Task 7. What is the output for the following program?**

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

class Node {
    Node *left, *right;
public:
    Node(){left = NULL; right = NULL;}
    ~Node(){}
    int process();
    Node** getLeft(){return &left;}
    Node** getRight(){return &right;}
};

int Node::process(){
    int retValue=0;
    if(left==NULL && right==NULL) return 1;
    if(left!=NULL) retValue += left->process();
    if(right!=NULL) retValue += right->process();
    return retValue;
}

class Level {
    int mLevel;
public:
    Level(int i);

    ~Level(){for (int i = 0; i < mLevel; i++) delete mNodes[i];}
    vector<Node*> mNodes;
    friend void link(Level& l1, Level& l2);
};

Level::Level(int i):mLevel(i){
    for(int j=0; j<mLevel;j++){
        mNodes.push_back(new Node());
    }
}

void link(Level& l1, Level& l2){
    for(int j=0;j<l2.mLevel;j++){
        *(l1.mNodes[j]->getRight()) = l2.mNodes[j];
        *(l1.mNodes[j+1]->getLeft()) = l2.mNodes[j];
    }
}

int main(int argc, char* argv[])
{
    Level* levels[12]; int i;
    for( i=0; i<12; i++)
        levels[i] = new Level(i);
    for( i=0;i<11;i++){
        link(*(levels[i+1]),*(levels[i]));
    }
    cout << levels[11]->mNodes[5]->process() << endl;

    for( i=0; i<12; i++)
        delete levels[i];
    return 0;
}
```



## Rešenja:

### Task1:

```
constructor Person
constructor Student
constructor PhDStudent
destructor Student
destructor Person
destructor Student
destructor Person
destructor Student
destructor Person
destructor Student
destructor Person
destructor PhDStudent
destructor Student
destructor Person
```

### Task2:

```
Base::f(double)
Derived::f(complex)
Base::f(double)
10
Derived::g() 20
Derived::g() 10
```

### Task3:

```
9.4
1.3
3.4
```

**Task4:**

Base A  
Base B  
Derived A  
Derived B  
Base B  
Base B  
Derived A  
Derived B  
Base B  
25000  
Base A  
Base B  
Base B  
625000000

**Task5:**

18

**Task6:**

64  
160  
96  
32  
208  
176  
144  
112  
80  
48  
16  
11

**Task7:**

252