

Object oriented programming – Elektrijska 2016 (Rimini)

Task1 (14):				Task5 (14):			
Task2 (14):				Task6 (15):			
Task3 (14):				Task7 (15):			
Task4 (14)							
Task1	Task2	Task3	Task4	Task5	Task6	Task7	Sum()

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

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

class Base {
protected:
    int x;
public:
    Base( int a ) : x(a) {};
    virtual int f( int );
    virtual double f( double );
    virtual int g( int i = 10 );
};

int Base::f( int a ) { cout << "Base::f(int)" << endl; return a+x; }

double Base::f( double a ) { cout << "Base::f(double)" << endl; return a+x/2; }

int Base::g( int i ) {
    if (i==10) { x = f(2*i); }
    cout << "Base::g() " << i << " " << x << endl;
    return x;
}

class Derived: public Base {
protected:
    int x;
public:
    Derived ( int a, int b ) : Base(a), x(b) {};
    double f( double );
    double f( complex<double> );
    int g( int i = 20 );
};

double Derived::f( double a ) { cout << "Derived::f(double)" << endl; return a+x/4; }

double Derived::f( complex<double> a ) { cout << "Derived::f(complex)" << endl; return
x/8; }

int Derived::g( int i ) {
    if (i==10) { x = f(2*i+1); }
    cout << "Derived::g() " << i << " " << x << endl;
    return x;
}

int f(Base &a) { return a.g(); }

class DerivedRimini : public Derived {
public:
    double f( complex<double> );
    DerivedRimini ( int a, int b ) : Derived(--a, ++b) {};
    int g( int i = 40 );
};

double DerivedRimini::f( complex<double> a ) { cout << "Derived::f(complex)" << endl;
return a.real()+x/16; }

int DerivedRimini::g( int i ) {
    if (i==10) { x = f(2*i+x); }
    if (i==40) { x = f(2*i+x+1); }
    cout << "DerivedRimini::g() " << i << " " << x << endl;
    return x;
}
```

```

class DerivedItaly : public DerivedRimini {
public:
    DerivedItaly ( int a, int b ) : DerivedRimini(++a, --b) {};
    int f( int );
    double f( double );
    int g( int i = 80 );
};

int DerivedItaly::f( int a ) { cout << "DerivedItaly::f(int)" << endl; return a+x*2; }

double DerivedItaly::f( double a ) { cout << "DerivedItaly::f(double)" << endl; return
a+x/8; }

int DerivedItaly::g( int i ) {
    if (i==10) { x = f(2*(i+x));}
    if (i==40) { x = f(2*(i+x)+1);}
    cout << "Derived::g() " << i << " " << x << endl;
    return x;
}

int main()
{
    Base b(2);
    Derived d(1,3);
    Base* pb = new Derived(2,4);
    DerivedRimini dr(3,5);
    DerivedRimini* pr = new DerivedItaly(4,8);
    DerivedItaly i(5,7);
    cout << d.f(b.g()) << endl;
    pb->f(1.0);
    d.g();
    cout << pb->g() << endl;
    cout << dr.f(f(i)) << endl;
    cout << pr->f(pr->g()) << endl;
    delete pb;
    delete pr;
    return 0;
}

```

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

```
#include <iostream>
#define BATTERY "economic"
#define BMB95 void

int cost = 0;
int itinirery = 3;
bool areWeThereYet(char* dest, int x = itinirery);

class Vehicle {
protected:
    int mileage;
public:
    Vehicle() : mileage(0) {
        //Is this brand new?\\
        std::cout << "new vehicle ";
        std::cout << "cost:" << (cost += 50) << std::endl;
    }
    Vehicle(int km) : mileage(km) {
        //This must be used vehicle...:/
        cost <= 1; std::cout << "cost:" << cost << std::endl; }
    void move(int distance) {
        std::cout << "Moved further to " << (itinirery % 2 ? "left" : "right") <<
std::endl;
        (itinirery % 2 ? cost : distance) += (cost % 3 ? 10 : -10);
        mileage += distance;
    }
};

void letsParty(Vehicle& vehicle);

class Engine {
public:
    Engine() { running = false; std::cout << "unknown engine cost:" << (cost *= 3) <<
std::endl; }
    Engine(char* type) { running = false; std::cout << type << " engine\n"; }
    void start() {
        if (running == false) {
            cost += 50;
            running = true;
            std::cout << "Brrrrmm\nstart cost:" << cost << std::endl;
        }
    }
    void stop() {
        if (running != false) {
            running = false;
            std::cout << "Cool-off\n";
        }
    }
    ~Engine() { stop(); }
private:
    bool running;
};

Vehicle bike;
class Car : public Vehicle {
    Engine e;
public:
    Car() { std::cout << "welcome" << std::endl; }
    Car(Engine engine) : Vehicle(100), e(engine) { std::cout << "awesome! cost:" <<
(cost - 50); }
```

```

int driveTo(char* location) {
    bool areWeThereYet(char* dest, int x = 10);
    e.start();
    while (!::areWeThereYet(location)) {
        move(50);
        std::cout << "mileage:" << mileage << " cost:" << cost << std::endl;
    }
    e.stop();
    return mileage;
}

};

int main() {
    Engine gasoline(BMB95), electric(BATTERY);
    Car yugo, tesla(electric);
    tesla.driveTo("Rimini");
    letsParty(yugo);
    return (int)0;
}

bool areWeThereYet(char* dest, int x) {
    itinirery--; std::cout << dest << ", wait for it..." << x << "\n";
    return x <= 0;
}

void letsParty(Vehicle& vehicle) {
    std::cout << "Budget:" << cost << " Andiamo, party! ";
}

```

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

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

class A {
public:
    A() {X=9;A1=A2=NULL;}
    A(int x) {X=x;A1=A2=NULL;}
    int X;
    A *A1, *A2;
    virtual void Drive();
};

void A::Drive() {
    cout << X * X << " ";
    if(A1!=NULL) A1->Drive();
    if(A2!=NULL) (*A2).Drive();
}

class B : public A {
public:
    B(int x):A(x) {A* t;t=A1;A1=A2;A2=t;}
    virtual void Drive();
};

void B::Drive() {
    if(A1!=NULL)A1->Drive();
    cout << X * X << " ";
    if(A2!=NULL)A2->Drive();
}

class C : public A {
public:
    C(int x):A(x){}
    virtual void Drive();
};

void C::Drive() {
    if(A2!=NULL)A2->Drive();
    if(A1!=NULL)A1->Drive();
    cout << X * X << " ";
}

int main() {
    A* a[10], *t = NULL;
    for(int i = 0; i<10; i++) {
        switch(i%3) {
            case(0):    a[i] = new A(i);    break;
            case(1):    a[i] = new B(i);    break;
            case(2):    a[i] = new C(i);    break;
            default:    a[i] = NULL;        break;
        }
        if(t==NULL) t = new A();
        switch(i%2)
        {
            case(1):    t->A2 = a[i];        t = a[i/2];        break;
            default:    t->A1 = a[i];        break;
        }
    }
    a[0]->Drive();
    return 0;
}
```

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

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int id = 0;
class A {
public:
    A() {
        id += objID = id *=-1;
    }
    A(int objectNumber) {
        id += objID = objectNumber;
        objID = id < 0 ? ++objID : -objID;
        cout << ++id << " OOP" << endl;
    }
    ~A() {
        id *=-1;
        id > 0 ? (cout << objID << " JADA " << endl, -id) : (id--, cout << -id << "
JADA " << endl, id);
    }
private:
    int objID;
};
class B : public A {
    A a;
public:
    B() : A(id) { }
    B(int i) : A(i) { static B b; }
    ~B() { A b(); }
};
class C : virtual public A {
public:
    C() : A(id) { }
    C(int i) : A(i) { static B b(i); }
    virtual ~C() { A b(1); }
};
class D : public B, public C {
public:
    D(int i) : B(1), C(i) { static A a(i); }
    ~D() { }
};
int main() {
    D b(id);
    exit(0);
}
B b(id++);
```

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

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

class Base{
protected:
    int x;
    int id;
public:
    Base() : id(1) { x = 16; cout << "Base 16" << endl;}
    Base(int xx) : x(++xx), id(1) { cout << "Base " << x << endl;}
    Base(Base& b) : x(b.x), id(1) { cout << "Base " << x << endl;}
    Base* fun(Base& b){
        b.x += x;
        return &b;
    }
    virtual Base funf(Base b){
        x += b.x;
        return *this;
    }
};

class Derived : public Base{
    Base bb;
public:
    Derived() { cout << "Derived 16" << endl;};
    Derived(int xx) : bb(8) { cout << "Derived " << x << endl;}
    Derived(Derived& d) : bb(8) { cout << "Derived " << x << endl;}
    Base* fun(Base& b){
        x = x << id + 1 | 0x1;
        return bb.fun(b);
    }
    Base funf(Base b){
        x = x << id + 1 | 0x2;
        bb.funf(b);
        return *this;
    }
    Derived funf(Derived b){
        x = x << 2 | 0x3;
        bb.funf(b);
        return *this;
    }
};

int main()
{
    Base b;
    Derived d;
    Base *bd = new Derived();
    Derived dd(4);

    Base *bf = b.fun(dd);
    b = bf->funf(b);
    bd->fun(dd);
    d = d.funf(dd);
    return 0;
}
```


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

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

template <class T>
class Node {
    T value, data;
    Node<T> *prev, *par, *next;
public:
    Node<T>(T v, Node *p, Node *pr, Node *n, T d=T()) {
        prev = p; par = pr; next = n;
        value = v; data = d;
    }
    Node<T>* Choose(T val, bool b=true) {
        Node<T>* ptr = this;
        for ( ; ptr!=NULL && (b && ptr->value!=val || !b && ptr->data!=val);
            ptr=ptr->next);
        return ptr;
    }
    Node<T>* Update(T val, T data) {
        Node<T>* ptr = this, *pptr = NULL;
        for ( ; ptr!=NULL && ptr->value<val; pptr=ptr, ptr=ptr->next);
        if (ptr==NULL || ptr->value != val) {
            Node *nNode = new Node(val, NULL, NULL, ptr, data);
            if (ptr == this) {
                ptr = nNode;
            } else {
                pptr->next = nNode;
                ptr = this;
            }
        }
        return ptr;
    }
    void UpdateData(T val, T data) {
        Node<T>* ptr = this;
        for ( ; ptr!=NULL && ptr->value!=val; ptr=ptr->next);
        bool bFind = false;
        if (ptr->prev == NULL) {
            ptr->prev = new Node(ptr->value, NULL, ptr, NULL, ptr->data);
            ptr->data = 1;
        } else {
            bFind = ptr->prev->Choose(data, false) != NULL;
        }
        if (!bFind) {
            ptr->data++;
            for (ptr=ptr->prev; ptr->next!=NULL; ptr=ptr->next);
            ptr->next = new Node(val, NULL, ptr->prev, NULL, data);
        }
    }
    void Change(Node<T>* node) {
        Node<T>* ptr = par;
        for ( ; ptr!=NULL && ptr->prev!=node; ptr=ptr->next);
        if (ptr == NULL && this != node) {
            par = new Node(0, node, NULL, par);
            node->par = new Node(0, this, NULL, node->par);
        }
    }
    void Print(Node<T>* node) {
        cout << value << " " << data << " -> ";
        Node<T>* ptr = node == NULL ? prev : par;
        for ( ; ptr!=NULL; ptr=ptr->next)
            cout << (node != NULL ? ptr->prev->value : ptr->prev->data) << " ";
        cout << endl;
        if (node != NULL && prev != NULL)
            prev->Print(NULL);
        if (node != NULL)
            cout << endl;
        if (next != NULL)
            next->Print(node);
    }
}
```

```

void UpdateData() {
    T start = T(1);
    Node<T>* ptr = prev;
    for ( ; ptr!=NULL; ptr=ptr->next) {
        T step = start;
        T val = ptr->data%10 + 1;
        for (Node<T>* pptr=ptr->next; pptr!=NULL; pptr=pptr->next) {
            for ( ; val < pptr->data%10+1; val+=step);
            if (val == pptr->data%10+1) {
                ptr->prev = new Node(0, pptr, NULL, ptr->prev);
                pptr->prev = new Node(0, ptr, NULL, pptr->prev);
            }
        }
        start = ptr->data%10 + 1;
    }
}

void CorrectData() {
    for (Node<T>* ptr=this; ptr!=NULL; ptr=ptr->next)
        ptr->UpdateData();
}

};

```

```

template <class T>
class Collection {
    Node<T>* major;
    void Update(T val, T data) {
        data = val * 10 + data - 1;
        Node<T>* ptr = major->Choose(val);
        if (ptr == NULL) {
            major = major->Update(val, data);
        } else {
            major->UpdateData(val, data);
        }
    }
    void Change(T val1, T val2) {
        Node<T>* ptr1 = major->Choose(val1);
        Node<T>* ptr2 = major->Choose(val2);
        ptr1->Change(ptr2);
    }
public:
    Collection() {
        major = NULL;
    }
    void Create(int n) {
        Update(1, 1);
        for (int i=2; i<=n; i++) {
            int j = 1;
            for (j=i/2; j>1 && i!=j*(i/j); j--);
            if (j != 1) {
                j = i/j;
            }
            Update(j, i);
            Update(i/j, i);
            Change(j, i/j);
        }
    }
    void UpdateData() {
        major->CorrectData();
    }
    void Print() {
        major->Print(major);
    }
};

```

```

int main() {
    Collection<int> coll;
    coll.Create(10);
    coll.UpdateData();
    coll.Print();
    return 0;
}

```

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

```
#include <iostream>
#include <math.h>
using namespace std;

template <class T>
class Elem {
    T a;
    T b;
public:
    Elem<T>() {
        a = T(); b = T();
    }
    Elem<T>(T aa, T bb) {
        a = aa;
        b = bb;
    }
    Elem<T> Join(Elem<T>& elj, bool bInc=false) {
        Elem<T> el(*this);
        if (bInc) {
            el.a += elj.a;
            el.b += elj.b;
        } else {
            el.a -= elj.a;
            el.b -= elj.b;
        }
        return el;
    }
    Elem<T> Move(T d) {
        Elem<T> el(d * a, d * b);
        return el;
    }
    T Evaluate(Elem<T>& pt) {
        T val = a * pt.a + b * pt.b;
        return val;
    }
    void Print() {
        cout << a << " " << b << endl;
    }
};

template <class T>
class Relation {
    Elem<T> min;
    Elem<T> max;
public:
    Relation<T>(Elem<T>& mini, Elem<T>& maxi) {
        min = mini; max = maxi;
    }
    double Check(Elem<T> el) {
        Elem<T> join = max.Join(min);
        Elem<T> elJoin = el.Join(min);
        T elVal = join.Evaluate(elJoin);
        if (elVal <= 0) {
            elVal = elJoin.Evaluate(elJoin);
        } else {
            T val = join.Evaluate(join);
            if (val <= elVal) {
                elJoin = el.Join(max);
                elVal = elJoin.Evaluate(elJoin);
            } else {
                T div = elVal / val;
                Elem<T> elNew = join.Move(div);
                elNew = min.Join(elNew, true);
                elJoin = elNew.Join(el);
                elVal = elJoin.Evaluate(elJoin);
            }
        }
        return sqrt(elVal);
    }
};
```

```

template <class T>
class Set{
    Elem<T>* arMem;
    int nMem;
    int* arSub;
    int nSub;
public:
    Set(int arV[], int nV) {
        nMem = nV;
        arMem = new Elem<T>[nMem];
        arSub = new int[nMem];
        for (int i=0; i<nMem; i++) {
            arMem[i] = Elem<T>((arV[i]&0xF0)>>4, arV[i]&0xF);
        }
        nSub = 0;
    }
    void CreateSub(int seed) {
        int sd = seed / 2;
        for (int i = sd; i<nMem; i+=seed) {
            arSub[nSub++] = i;
        }
        arSub[nSub++] = sd;
    }
    void UpdateSub(double e) {
        for (int s=0,c=0; s<nSub-1; s+=c+1) {
            c = Check(s, arSub[s], arSub[s+1], e);
        }
    }
    int Check(int m, int s, int ss, T e) {
        int check = 0;
        Relation<T> rel(arMem[s],arMem[ss]);
        int belongs = s;
        T dBelongs = e;
        for (int i=(s+1)%nMem; i!=ss; i=(i+1)%nMem) {
            T dB = rel.Check(arMem[i]);
            if (dB > dBelongs) {
                dBelongs = dB;
                belongs = i;
            }
        }
        if (dBelongs != e) {
            for (int i = nSub++; i>m; i--) {
                arSub[i] = arSub[i-1];
            }
            arSub[m+1] = belongs;
            check += Check(m+1, belongs, ss, e);
            check += Check(m, s, belongs, e);
            check ++;
        }
        return check;
    }
    void Print() {
        for (int i=0; i<nSub; i++) {
            arMem[arSub[i]].Print();
        }
    }
};

int main() {
    int nVal = 0x1E;
    int arVal[] = {0x84, 0x73, 0x51, 0x42, 0x23, 0x12, 0x2, 0x33, 0x26, 0x35,
                   0x28, 0x47, 0x56, 0x79, 0x9D, 0xAE, 0xAF, 0xBD, 0xCA, 0xC8,
                   0xB6, 0xD7, 0xD6, 0xE5, 0xE3, 0xF2, 0xE3, 0xC2, 0xA3, 0x95};
    Set<double> set(arVal, nVal);
    set.CreateSub(0xA);
    set.UpdateSub(2);
    set.Print();
    return 0;
}

```

Solutions: Object oriented programming – Rimini 2016

Task1.

```
Base::f(int)
Base::g() 10 22
Derived::f(double)
22
Derived::f(double)
Derived::g() 20 3
Derived::f(double)
Derived::g() 10 22
22
DerivedItaly::f(int)
Derived::g() 10 48
Derived::f(complex)
48
DerivedItaly::f(int)
Derived::g() 40 113
Derived::f(complex)
120
```

Task2.

```
cost:50
economic engine
cost:100
unknown engine cost:300
welcome
cost:600
awesome! cost:550Cool-off
Brrrrmm
start cost:650
Rimini, wait for it...3
Moved further to right
mileage:160 cost:650
Rimini, wait for it...2
Moved further to left
mileage:210 cost:660
Rimini, wait for it...1
Moved further to right
mileage:250 cost:660
Rimini, wait for it...0
Cool-off
Budget:660 Andiamo,
party! Cool-off
Cool-off
Cool-off
```

Task3.

```
0 49 36 4 9 64 81
```

Task4.

```
2 OOP
-7 OOP
-26 OOP
67 OOP
-119 OOP
15 JADA
120 JADA
-14 JADA
121 JADA
0 JADA
122 JADA
-3 JADA
```

Task5.

```
Base 16
Base 16
Base 16
Derived 16
Base 16
Base 16
Derived 16
Base 16
Base 9
Derived 16
Base 16
Base 16
Base 25
Base 130
Base 16
Base 9
Derived 16
Base 16
Base 32
Base 16
Base 9
Derived 16
```

Task6.

```
1 5 -> 7 5 3 2
1 10 -> 16 14 12 11
1 11 -> 16 14 12 10
1 12 -> 16 14 11 10
1 14 -> 12 11 10
1 16 -> 12 11 10

2 5 -> 5 4 3 1
2 21 -> 29 27 25 23
2 23 -> 29 27 25 21
2 25 -> 29 23 21
2 27 -> 23 21
2 29 -> 25 23 21
```

```
3 3 -> 2 1
3 32 -> 38 35
3 35 -> 38 32
3 38 -> 35 32
```

```
4 47 -> 2
```

```
5 2 -> 2 1
5 54 -> 59
5 59 -> 54
```

```
7 76 -> 1
```

Task7.

```
1 2
2 8
5 6
10 14
11 6
13 7
15 2
9 5
5 1
1 2
```