

Object oriented programming – Elektrijska 2017 (Budva)

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>
using namespace std;

class A
{
protected:
    int ax;
    virtual void f(A a = 2) { ax+=a.ax; }
public:
    A() { ax = 0; }
    A(int a) { ax = a; }

    bool operator!=(A& a) { return ax != a.ax; }

    A& operator++() { cout << "Becici++ " << ax+++2 << endl; return *this; }
    A& operator++(int) { cout << "Budva++ " << ax+++2 << endl; return *this; }
    A& operator--() { cout << "Becici-- " << ax---2 << endl; return *this; }
    A& operator--(int) { cout << "Budva-- " << ax---2 << endl; return *this; }

    A& operator+() { f(); cout << "opA+() " << ax << endl; return *this; }
    A& operator+(int i) { f(i); cout << "opA+(int) " << ax << endl; return *this; }
    A& operator+(A& a) { cout << "opA+(A) " << ax << endl; return *this; }
    A& operator-() { f(); cout << "opA-() " << ax << endl; return *this; }
    A& operator-(int i) { f(i); cout << "opA-(int) " << ax << endl; return *this; }
    A& operator-(A& a) { cout << "opA-(A) " << ax << endl; return *this; }

    virtual A& operator=(A &a) {
        cout << "A::op= " << ax << endl;
        ax = a.ax;
        return *this;
    };
};

const A aa(4);
class B : public A
{
    int bx;
    void f(A a = aa) { A::f(a); bx<=1; }
public:
    B() : A(aa) { bx = ax; }
    B(int a) : A(a) { bx = ax / 2; }

    A& operator++() {
        cout << "Becici++ " << ax+++4 << " " << bx+++4 << endl; return *this; }
    A& operator++(int) {
        cout << "Budva++ " << ax+++4 << " " << bx+++4 << endl; return *this; }
    A& operator--() {
        cout << "Becici-- " << ax---4 << " " << bx---4 << endl; return *this; }
    A& operator--(int) {
        cout << "Budva-- " << ax---4 << " " << bx---4 << endl; return *this; }

    B& operator+() {
        f(); cout << "opB+() " << ax+bx << endl; return *this; }
    B& operator+(int i) {
        f(i); cout << "opB+(int) " << ax+bx << endl; return *this; }
    B& operator+(B& b) {
        f(b); cout << "opB+(B) " << ax+bx << endl; return *this; }
    A& operator-() {
        f(); cout << "opB-() " << ax + bx << endl; return *this; }
    A& operator-(int i) {
        f(i); cout << "opB-(int) " << ax + bx << endl; return *this; }
    A& operator-(B& b) {
        f(b); cout << "opB-(B) " << ax + bx << endl; return *this; }
```

```

A& operator =(A &a)
{
    if (*this != a)
        a = *this;
    bx = bx / ax / 2;
    cout << "B::op= " << ax << endl;
    return *this;
};

};

A& operator+(A& a) { cout << "op+(A)ext" << endl; return a; }
A& operator+(int g, A& a) { cout << "op+(ul,A)ext" << endl; return a; }
A& operator-(A& a) { cout << "op-(A)ext" << endl; return a; }
A& operator-(int g, A& a) { cout << "op-(g,A)ext" << endl; return a; }

int main()
{
    int x = 7;
    A a1, a2(3);
    B b1(5), b2;
    A a3 = a1+++++b1-a2+b2;
    A &ra = b2 = a1+++++b2+x---a3;
    B &rb = b1 = b1+x+b2;
    B bb = rb+x+++b1+b2;
    return 0;
}

```

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

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

class A {
public:
    A() : f(false) {
        cout << "cons A" << endl;
    }
    A(int objectNumber) : f(true) {
        // Little-known operator "-->", also known as "goes to"
        while( objectNumber --> 0 ) {
            cout << (objectNumber --> 0) << " ";
        }
        cout << "cons A(int)" << endl;
    }
    bool f;
};

class B : virtual private A {
    A a;
public:
    B(int i) : A(i) { B b(); }
    ~B() { static A b; cout << "dest B" << (f ? "(int)" : "") << endl; }
    B() : a(1) { }
};

class C : public A {
public:
    C() : A(0) { }
    C(int i) : A(i) { static B b; }
    virtual ~C() { A b(1); cout << "dest C" << (f ? "(int)" : "") << endl; }
};

class D : public B, virtual public C, virtual public A {
public:
    D(int i) : B(2), C(i) { static A a(i); }
};

int main() {
    D b(2);
}
```

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

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

class A
{
public:
    A() { Val = 9; x = y = z = NULL; }
    A(int w) { Val = w; x = y = z = NULL; }
    int Val;
    A *x;
    A *y;
    A *z;
    virtual void Drive();
    string Iterate(int i);
    friend ostream& operator<<(ostream& os, A& a);
};

void A::Drive()
{
    cout << *this << endl;
    if (x != NULL) x->Drive();
    if (y != NULL) (*y).Drive();
    if (z != NULL) z->Drive();
}

string A::Iterate(int i) {
    string s;
    if (i > 0)
    {
        s += Iterate(i / 2);
        s += (i & 0x1) | 0x30;
    }
    return s;
};

ostream& operator<<(ostream& os, A& a)
{
    os << setfill('0') << setw(4) << a.Iterate(a.Val);
    return os;
}

class B : public A
{
public:
    B(int w) :A(w) { A* t; t = x; x = y; y = z; z = t; }
    virtual void Drive();
};

void B::Drive()
{
    if (x != NULL)x->Drive();
    cout << *this << endl;
    if (y != NULL)y->Drive();
}
```

```

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

void C::Drive()
{
    if (z != NULL)y->Drive();
    if (y != NULL)x->Drive();
    cout << *this << endl;
}

int main(int argc, char* argv[])
{
    A* a[15], *t = NULL;
    for (int i = 0; i<15; 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 % 5)
        {
            case(1):    t->y = a[i];    t = a[i / 2];    break;
            case(2):    t->x = a[i];    t = a[i / 3];    break;
            case(3):    t->y = a[i];    t = a[i / 4];    break;
            case(4):    t->x = a[i];    t = a[i / 5];    break;
            case(5):    t->y = a[i];    t = a[i / 2];    break;
            default:    t->z = a[i];    break;
        }
    }

    a[0]->Drive();
}

```

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

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

#define TRUE  '///'
#define FALSE '-'-'-
#define zero FALSE

static int id = 0;

class A {
    int info;
public:
    A() { info = 0; cout << "A(default)" << endl; id = 1; }
    A(double x) { info = x; cout << "A(double) " << x * 2 << endl; id++; }
    A(const A& x) { info = x.info; cout << "A(A) " << (id += 5) << endl; }
    A& operator=(double x) { info += x; return *this; }
    friend ostream& operator<<(ostream& os, A& x) {
        os << x.info << " "; return os;
    }
};

A a = 5.7;
void f(int x) { cout << "f(int) " << x << endl; };
void f(A& x) { cout << "f(A)" << endl; };

void g(A x) {
    cout << x << (id + 2.0) << " ";
    A obj = id + id;
}

int main() {
    int i = TRUE;
    A a;

    f(i ? 1 : 2.5);
    f(i ? 2.5 : 1);

    int x = 2 * TRUE * 2;
    while ( x --> zero) // x goes to zero
    {
        extern A a;
        a = x;
        g(a);
    }

    zero;
    x = 2 * FALSE * 2;

    cout << x << endl;
    return 0;
}
```

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

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

class Base {
protected:
    int x;
    int bv;
public:
    Base() : x(2), bv(-1) { }
    Base(int v) : x(2), bv(v) { }
    virtual bool operator <(Base& b) {
        if (this == NULL) throw b;
        if (&b == NULL || bv < -1) throw this;
        return bv < b.bv;
    }
    bool operator >=(Base& b) { return !(*this < b); }
    virtual int check(float f) {
        if ((int)(x*f) != x*f) throw f;
        return (int)(x*f);
    }
    virtual int check(double d) {
        if ((int)(x*x*d) != x*x*d) throw d;
        return (int)(x*x*d);
    }
    bool operator &&(Base& b) {
        if (this == NULL) throw b;
        if (&b == NULL) throw this;
        return this->bv && b.bv;
    }
};

class Derived : Base {
    int dv;
public:
    Derived() : dv(-1) { }
    Derived(int v, int vv) : Base(v), dv(vv) { }
    bool operator <(Base& b) {
        if (this == NULL) throw b;
        if (&b == NULL) throw this;
        if (dv < 0) throw &b;
        return b >= *this;
    }
    bool operator >=(Base& b) { return !(*this < b); }
    virtual int check(double d) {
        if ((int)(x*x*d) != x*x*d) throw d;
        return (int)(x*x*d);
    }
    Base* convert(float f, double d) {
        if (f < d) return new Base(check(f));
        else if (f > d) return new Derived(check(f), check(d));
        else return NULL;
    }
    int prepare(float arf[], double ard[], int& iter, int limit) {
        Base b;
        Base *pb = NULL, *pb2 = NULL, *pb3 = NULL;
        try {
            while (iter < limit && pb3 == NULL) {
                cout << iter << " ";
                if (pb == NULL || pb2 == NULL) {
                    if (pb == NULL)
                        pb = convert(arf[iter], ard[iter]);
                    if (*pb && b) {
                        Base bb(1);

```



```

        if (*pb < bb)
            throw pb;
    }
}
else {
    pb3 = convert(arf[iter], ard[iter]);
    if (*pb3 && b) {
        iter <= 8;
        cout << endl << "finish!" << endl;
        return iter;
    }
}
if (pb2 == NULL) {
    pb2 = pb;
    pb = NULL;
}
iter++;
cout << "round!" << endl;
}

}
catch (Base) { cout << "Base "; }
catch (double) { cout << "double "; }
catch (Derived) { cout << "Derived "; }
catch (float) { cout << "float "; }
catch (Derived*) { cout << "Derived* "; }
iter++;
cout << endl;
return iter;
}

};

int main() {
    float arf[8] = { 2.375, 5, 4.5, 2.25, -5.75, 0, 9.75 };
    double ard[8] = { 3.25, 3.625, 4.5, 3.75, -7.5, -0.5, 4.5 };
    int i = 0, n = 8;
    Derived d;
    while (i < n) {
        try {
            d.prepare(arf, ard, i, n);
            i--;
        }
        catch (Derived&) { cout << "Derived& " << endl; }
        catch (double) { cout << "double " << endl; }
        catch (Base&) { cout << "Base& " << endl; }
        catch (float) { cout << "float " << endl; }
        catch (Base*) { cout << "Base* " << endl; }
        catch (...) { cout << "..." << endl; }
        i++;
    }
    return 0;
}

```

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

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

template <class Tip> class Point
{
public:
    Tip x, y;
    Point() { x = 0; y = 0; }
    Point(Tip xx, Tip yy) { x = xx; y = yy; }
    bool operator <(Tip t) { return x + y < t; }
    Tip Sum() { return x + y; }
};

template <class Tip> class Item
{
public:
    Point<Tip> key, value;
    Item *par, *link;
    Item(Point<Tip> k, Point<Tip> v, Item<Tip>* n) {
        key = k; value = v;
        par = NULL; link = n;
        if (link != NULL) link->par = this;
    }
    bool Compare(Item<Tip> *ptr) {
        return key.y + value.y < ptr->key.y;
    }
    bool Relate(Item<Tip> *ptr) {
        return key.y < ptr->key.y + ptr->value.y && key.y + value.y > ptr->key.y;
    }
    Tip Value() {
        cout << key.x << " " << key.y << " " << value.y << endl;
        return (link != NULL) ? value.y + link->Value() : value.y;
    }
};

template <class Tip, int n> class Container
{
    Point<Tip> **arPoint;
    Item<Tip> **arItems;
    Item<Tip> **arData;
    void Insert(Point<Tip> k, Point<Tip> v)
    {
        arItems[k.x] = new Item<Tip>(k, v, arItems[k.x]);
    }
    void Delete(int ind, Item<Tip>* ptr)
    {
        if (ptr->par == NULL)
            arItems[ind] = arItems[ind]->link;
        else
            ptr->par->link = ptr->link;
        if (ptr->link != NULL)
            ptr->link->par = ptr->par;
    }
    void Process(int iter, int ind, Item<Tip> *item)
    {
        arData[iter] = new Item<Tip>(item->key, item->value, arData[iter]);
        Delete(ind, item);
        for (int i = -1; i <= 1; i += 2) {
            if (ind + i >= 0 && ind + i < n) {
                Item<Tip> *ptr = arItems[ind + i];
                for (; ptr != NULL && item->Compare(ptr); ptr = ptr->link);
                while (ptr != NULL && item->Relate(ptr)) {
                    Process(iter, ind + i, ptr);
                    for (ptr = arItems[ind + i]; ptr != NULL && item->Compare(ptr);
                        ptr = ptr->link);
                }
            }
        }
        delete item;
    }
};
```

```

public:
    Container(Tip *arr)
    {
        int div = 4;
        arPoint = new Point<Tip>*[n];
        for (int i = 0; i < n; i++) {
            arPoint[i] = new Point<Tip>[n];
            unsigned int shift = 0x18;
            unsigned int mask = 0xF << shift;
            for (int ii = 0; ii < n; ii++) {
                arPoint[i][ii] = Point<Tip>((arr[(i*n + ii) / div] & (mask << 4)) >> (shift + 4),
                    (arr[(i*n + ii) / div] & mask) >> shift);
                shift -= shift > 0 ? 0x8 : -0x18;
                mask = mask > shift ? mask >> 0x8 : 0xF << shift;
            }
        }
    }
    void Prepare(Tip t)
    {
        arItems = new Item<Tip>*[n];
        arData = new Item<Tip>*[n];
        for (int i = 0; i < n; i++)
            arItems[i] = arData[i] = NULL;
        for (int i = 0; i < n; i++) {
            int ii = 0;
            while (ii < n) {
                for (; ii < n && arPoint[i][ii] < t; ii++);
                int iii = ii;
                for (; ii < n && !(arPoint[i][ii] < t); ii++);
                if (iii < ii)
                    Insert(Point<Tip>(i, iii), Point<Tip>(arPoint[i][ii].Sum(), ii- iii));
            }
        }
    }
    void Process()
    {
        int iter = -1;
        for (int i = 0; i < n; i++)
            while (arItems[i] != NULL)
                Process(++iter, i, arItems[i]);
        for (int i = 0; i <= iter; i++)
            cout << arData[i]->Value() << endl;
    }
};

int main() {
    unsigned int arr[] = {
        0xb87bca95, 0x87a6f3b7, 0xf4fadbd4, 0xd9c8b7e8,
        0xd5dceeda, 0xc6e3d9f8, 0xe4b7fcc9, 0xbad8e7f4,
        0xc59887b6, 0xa8c7d4d6, 0xe8f7c4cc, 0xe9dbeac8,
        0xc9a6d8eb, 0xb6d3f9e5, 0xb489a8e8, 0xdae9cbc4 };
    Container<unsigned int, 0x8> container(arr);
    container.Prepare(0x14);
    container.Process();
    return 0;
}

```

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

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

template <class Tip> class Element
{
public:
    Tip key;
    Tip value;
    Element<Tip> *data, *prev, *next;
    Element<Tip>(Tip k, Tip v, Element<Tip>* d, Element<Tip>* p, Element<Tip>* n) {
        key = k; value = v;
        prev = p; next = n; data = d;
        if (p != NULL) prev->next = this;
        if (n != NULL) next->prev = this;
    }
    void Visit() { cout << key << " " << value << endl; }
    void Update(int v, Element<Tip>* p) { value = v; prev = p; }
};

template <class Tip> class Collection
{
    Element<Tip> *first;
public:
    Collection() { first = NULL; }
    bool IsEmpty() { return first == NULL; }
    void Insert(Element<Tip>* data) {
        if (first != NULL) {
            first = new Element<Tip>(0, 0, data, first, first->next);
        } else {
            first = new Element<Tip>(0, 0, data, first, first);
            first->next = first->prev = first;
        }
    }
    Element<Tip>* Delete() {
        Element<Tip>* ptr = first->next;
        if (first->next != first) {
            first->next = first->next->next;
            first->next->prev = first;
        } else {
            first = NULL;
        }
        return ptr->data;
    }
};

template <class Tip> class Connection
{
    Element<Tip>* start;
    int count;
    Element<Tip>* Find(int key) {
        Element<Tip>* ptr = start;
        while (ptr != NULL && ptr->key != key)
            ptr = ptr->next;
        return ptr;
    }
    Element<Tip>* Insert(int key) {
        Element<Tip>* ptr = Find(key);
        if (ptr == NULL)
            ptr = start = new Element<Tip>(key, ++count, NULL, NULL, start);
        return ptr;
    }
    Element<Tip>* Find(Element<Tip>* ptr1, Element<Tip>* ptr2) {
        Element<Tip>* data = ptr1->data;
        while (data != NULL && data->data != ptr2)
            data = data->next;
        return data;
    }
}
```

```

void Insert(Element<Tip>* ptr1, Element<Tip>* ptr2, int key) {
    Element<Tip>* ptr = Find(ptr1, ptr2);
    if (ptr == NULL)
        ptr1->data = new Element<Tip>(key, 0, ptr2, NULL, ptr1->data);
}

void Update(Element<Tip>* ptr1, Element<Tip>* ptr2, int value) {
    Element<Tip>* ptr = Find(ptr1, ptr2);
    if (ptr != NULL) ptr->value += value;
}

public:
    Connection<Tip>(int *map, int size) {
        start = NULL;
        count = 0;
        for (int i = 0; i < size; i++) {
            Element<Tip> *ptr1 = Insert(map[3 * i]);
            Element<Tip> *ptr2 = Insert(map[3 * i + 1]);
            Insert(ptr1, ptr2, map[3 * i + 2]);
            Insert(ptr2, ptr1, 0);
        }
    }

    long Refresh(Element<Tip> *mini, Element<Tip> *maxi) {
        Element<Tip>* ptr = start;
        for (; ptr != NULL; ptr = ptr->next)
            ptr->prev = NULL;
        ptr = mini;
        Collection<Tip> coll;
        coll.Insert(ptr);
        ptr->Update(17, ptr);
        while (!coll.IsEmpty() && ptr != maxi) {
            ptr = coll.Delete();
            if (ptr != maxi) {
                Element<Tip>* elem = ptr->data;
                while (elem != NULL) {
                    if (elem->data->prev == NULL && elem->key - elem->value > 0) {
                        elem->data->Update(min(ptr->value, elem->key - elem->value),
                                           ptr);
                        coll.Insert(elem->data);
                    }
                    elem = elem->next;
                }
            }
        }
        return ptr == maxi ? ptr->value : 0;
    }

    int Process(int minValue, int maxValue) {
        int count = 0;
        Element<Tip>* min = Find(minValue);
        Element<Tip>* max = Find(maxValue);
        int val = 0;
        while ((val = Refresh(min, max)) != 0) {
            Element<Tip>* ptr = max;
            while (ptr != min) {
                ptr->Visit();
                Element<Tip>* pptr = ptr->prev;
                Update(pptr, ptr, val);
                Update(ptr, pptr, -val);
                ptr = pptr;
            }
            ptr->Visit();
            count += val;
        }
        return count;
    }
};

int main() {
    int arr[] = { 1,2,10, 1,3,5, 1,4,15, 2,3,4, 2,5,9, 2,6,8, 3,4,4,
                  3,6,8, 4,7,15, 5,6,5, 5,8,11, 6,7,10, 6,8,12, 7,3,6, 7,8,10 };
    Connection<int> connection(arr, 15);
    cout << connection.Process(1, 8);
    return 0;
}

```

Answers: Object oriented programming – Elektrijska 2017 (Budva)

0101

5 0 2
3

Task1.

Budva++ 2
Budva++ 3
opA+ (A) 2
opA- (A) 2
opA+ (A) 2
Budva++ 4
Budva++ 5
opA+ (A) 4
opA+ (int) 11
opA- (A) 11
A::op= 11
B::op= 4
opB+ (int) 15
opB+ (B) 23
A::op= 15
opB+ (int) 37
opB+ (B) 74
opB+ (B) 110

Task2.

cons A
1 cons A(int)
cons A
0 cons A(int)
cons A
1 cons A(int)
cons A
dest B
0 cons A(int)
dest C(int)
dest B

Task3.

0000
0100
1101
0110

Task4.

A(double) 11.4
A(default)
f(int) 1
f(int) 2
A(A) 6
8 8 A(double) 24
A(A) 12
10 14 A(double) 48
A(A) 18
11 20 A(double) 72
A(A) 24
11 26 A(double) 96
0

Task5.

0 double
1 double
2 Base
3 round!
4 Base*
5 round!
6 round!
7 Base

Task6.

1 7 1
2 6 2
3 2 5
2 1 3
1 1 2
0 2 1
14
1 4 2
2
6 2 2
7 3 4
6 6 1
5 3 5
12
6 0 1

Task7.

8 10
7 15
4 15
1 17
8 5
6 5
3 5
1 17
8 7
6 8
2 10
1 17
8 3
5 3
2 3
1 17
8 3
5 3
2 3
6 3
3 5
7 5
4 5
1 17
28