

Object oriented programming – Elektrijska 2023 (Ohrid)

Task 1 (14):

Task 2 (14):

Task 3 (14):

T.1. (14)	T.2. (14)	T.3. (14)	T.4. (14)	T.5. (14)	T.6. (14)	T.7. (16)	Total

Task 4 (14):
Task 5 (14):
Task 6 (14):
Task 7 (16):

Object oriented programming – Elektrijada 2023 (Ohrid)

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

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

using F = function<int>();

class A {
public:
    int v;
    F f = [&]() { cout << "A::f" << endl; return v += ff(); };

    A() : v(3) { cout << "A()" << endl; }
    A(int vv) : v(vv) { cout << "A(" << vv << ")" << endl; }
    A(const A& a) : v(a.v / 2) { cout << "A(A(" << a.v << "))" << endl; }
    virtual ~A() { cout << "~A(" << v << ")" << endl; }

    virtual int ff() const { cout << "A::ff()c "; return v + 1; }
    virtual int ff() { cout << "A::ff() "; return v += 1; }
};

struct B : virtual A {
    const F fa = A::f;
    F f = [&]() { cout << "B::f" << endl; return v *= ff(); };

    B() : A(2) { cout << "B()" << endl; }
    B(const B& b) { cout << "B(B())" << endl; ++++++v; }
    virtual ~B() { cout << "~B()" << endl; }

    virtual int ff() const { cout << "B::ff()c "; return v + 2; }
    virtual int ff() { cout << "B::ff() "; return v += 2; }
};

int main() {
    A a;
    B b;

    cout << a.f() << endl;
    cout << b.f() << endl;
    cout << b.fa() << endl;
    cout << "A = " << a.v << " & B = " << b.v << endl;

    [=]() {
        const A* a = &b;

        cout << a->f() << endl;
        cout << a->ff() << endl;

        cout << "A = " << a->v << endl;

        cout << b.f() << endl;
        cout << b.ff() << endl;

        cout << "B = " << b.v << endl;
    }();

    cout << "A = " << a.v << " & B = " << b.v << endl;
    return 0;
}
```

Task 2. 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; }
    bool operator<(A& a) { return ax < a.ax; }
    A& operator++() { cout << "Jezero++ " << ax++*2 << endl; return *this; }
    A& operator++(int) { cout << "Ohrid++ " << ax++*2 << endl; return *this; }
    A& operator--() { cout << "Jezero-- " << ax--*2 << endl; return *this; }
    A& operator--(int) { cout << "Ohrid-- " << 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 << "Jezero++ " << ax++*4 << " " << bx++*4 << endl; return *this;
    }
    A& operator++(int) {
        cout << "Ohrid++ " << ax++*4 << " " << bx++*4 << endl; return *this;
    }
    A& operator--() {
        cout << "Jezero-- " << ax--*4 << " " << bx--*4 << endl; return *this;
    }
    A& operator--(int) {
        cout << "Ohrid-- " << 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;
    }
    B& operator=(B a) {
        cout << "B::op= " << ax << " " << bx << endl; return *this;
    };
};

class C : public A {
    int cx;
    void f(A a = aa) { A::f(a); cx <= 2; }
public:
    C() : A(aa) { cx = ax; }
    C(int a) : A(a) { cx = ax / 4; }
    C& operator+() {
        f(); cout << "opC+()" << ax + cx << endl; return *this;
    }
    C& operator+(int i) {
        f(i); cout << "opC+(int) " << ax + cx << endl; return *this;
    }
    C& operator+(C& c) {
        f(c); cout << "opC+(B) " << ax + cx << endl; return *this;
    }
    C& operator =(C a) {
        cout << "C::op= " << ax << " " << cx << endl; return *this;
    };
};

A& operator+(A& a) { cout << "op+(A)ext" << endl; return a; }
int& operator+(int& g, A& a) { cout << "op+(u1,A)ext" << endl; return g; }
A& operator-(A& a) { cout << "op-(A)ext" << endl; return a; }
int& operator-(int& g, A& a) { cout << "op-(g,A)ext" << endl; return g; }

A& max(A* pa1, A* pa2)
{
    cout << "max(a1,a2)" << endl;
    return *pa1 < *pa2 ? *pa2 : *pa1;
}

int main()
{
    int x = 7;
    A a1, a2(3);
    B b[3];
    C c[3];
    A& ra1=b[1]=b[0]+x+b[1];
    B b3=b[2]=++x+a1+++++a2+a2;
    A& ra2=c[0]+c[1]=c[0]+x+c[1];
    ra1 = ra1 < b[2] ? ra1 : b[2];
    ra2 = ra2 < c[2] ? ra2 : c[2];
    ra1 = ra1 < ra2 ? ra1 : ra2;
    A* pb = max(&b[1], &b[2]);
    C* pc = max(&c[1], &c[2]);
    *pb = max(pb, pc);
    return 0;
}

```

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

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

class A
{
public:
    static int id;
    A(int objNum=id+1) {
        id += objID = objNum;
        cout << "consA " << id << endl;
    }
    ~A() {
        cout << "--id << " dest " << endl;
    }
    virtual void f(int n = 1) {
        cout << "A::f->" << objID + n << endl;
    }
protected:
    int objID;
};

int A::id = 0;

class B : virtual public A
{
public:
    B() : A(id) {
        cout << (id >= 2) << " consB" << endl;
    }
    B(int i) : A(i) {
        cout << "consB " << id << endl;
    }
};

class C : virtual public A {
    A a;
public:
    C() : A(id) {
        cout << (id *= 2) << " consC" << endl;
    }
    C(int i) : A(i) { cout << "consC " << id << endl; }
    C(const C& e) { cout << "consC " << id << endl; }
};

class D : public B, virtual public C {
public:
    D(int i = 0) : C(id) {
        cout << "consD " << id << endl;
    }
    void f(int n = 2) {
        cout << "D::f->" << objID + n << endl;
    }
};

class E : public D {
private:
    void f(int n = 4) {
        cout << "E::f->" << objID + n << endl;
    }
};
```

```

class F : public B, public E {
    D d;
public:
    F(int i) : B(id), d(i), E() {
        B b(i);
        cout << "consF " << id << endl;
    }
    F(const E& e) {
        cout << "consF " << id << endl;
    }
    void f(int n = 8) {
        cout << "F::f->" << objID + n << endl;
    }
};

int main() {
    F f(0);
    F ff = f;
    C& rc = f;
    D* pd = &f;
    A* pa = &rc;
    rc.f();
    pd->f();
    pa->f();
    exit(0);
}

```

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

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

class Base
{
protected:
    int a;
public:
    Base(int aa = 2) : a(aa) {
        cout << "Base " << a << endl;
    }
    Base& operator =(const int& aa) {
        a = aa + 1;
        cout << "Base= " << a << endl;
        return *this;
    }
    virtual int f(int x) {
        x = x << a + 1 | a;
        cout << "Bf " << x << endl;
        return x;
    }
    virtual void g(Base& b, Base x=1) {
        cout << "Bg->";
        b.f(x.a);
    }
};

class Derived : public Base
{
public:
    Derived(int aa) {
        cout << "Der " << a << endl;
    }
    Base& operator =(const int& aa) {
        cout << "Der::";
        return Base::operator=(aa);
    }
    int f(int x) {
        x = x << a + 1 | a;
        cout << "Df " << x << endl;
        return x;
    }
private:
    void g(Base& b, Base x) {
        cout << "Dg->";
        b.f(1);
    }
};

class Particular : public Derived
{
    Base b;
    Derived d;
public:
    Particular(int aa) : Derived(aa), d(aa) {
        cout << "Part " << a << endl;
    }
    Base& operator =(const int& aa) {
        b = d = aa + 1;
        cout << "Part::";
        return Derived::operator=(aa);
    }
}
```



```

    int f(int x) {
        x = x >> a + 1 | a;
        cout << "Pf " << x << endl;
        return x;
    }
    void g(Base& b, Base x) {
        cout << "Bg->";
        b.f(1);
    }
};

int main()
{
    int x = 1;
    Base b = 1;
    Derived d = 3;
    Particular p = 5;
    Base* pbd = &d;
    Base* pbp = &p;
    Derived* pdp = &p;

    x = b.f(x);
    x = p.f(x);
    x = pbd->f(x);
    x = pdp->f(x);
    x = pbp->f(x);

    b.g(b);
    pbd->g(d);
    pbp->g(p);

    b.g(b, x);
    pbd->g(d, x);
    pbp->g(p, x);

    return 0;
}

```

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

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

class Data {
    int val, key;
public:
    Data() : key(), val() { }
    Data(int k, int v) : key(k), val(v) { }
    friend ostream& operator<<(ostream& out, Data& b) {
        out << b.key << " " << b.val;
        return out;
    }
};

template <class T, class U>
class Derived;

template <class T, class U>
class Base {
protected:
    T key;
    U val;
public:
    Base() : key(), val() { }
    Base(T k, U v) : key(k), val(v) { }
    Base(const Base<T, U>& b) : key(b.key), val(b.val) { }
    bool operator<(Base<T, U>& b) {
        return val < b.val;
    }
    void Update(Base<T, U>& e1, Base<T, U>& e2) {
        val += e1 < e2 ? e1.val : e2.val;
    }
    friend ostream& operator<<(ostream& out, Base<T, U>& b) {
        out << b.key << " " << b.val << endl;
        return out;
    }
    virtual void f(Base<T, U> e1, Base<T, U> e2) {
        Update(e1, e2);
        cout << "fb1 " << *this;
    }
    virtual void f(Base<T, U> e1, Derived<T, U>& e2) {
        Update(e1, e2);
        cout << "fb2 " << *this;
    }
    virtual void f(T v1, T v2) {
        cout << "fb3 " << *this;
    }
    template <class V>
    void f(V e1, V e2) {
        Update(e1, e2);
        cout << "fb4 " << *this;
    }
    void f(T) {
        cout << "fb5 " << *this;
    }
    template<class V>
    void f(V*) {
        cout << "fb6 " << *this;
    }
};
```

```

template <class T, class U>
class Derived : public Base<T, U> {
public:
    Derived() : Base<T, U>() { }
    Derived(T k, U v) : Base<T, U>(k, v) { }
    void f(Base<T, U> e1, Base<T, U> e2) {
        Base<T, U>::Update(e1, e2);
        cout << "fd1 " << *this;
    }
    virtual void f(T v1, T v2) {
        cout << "fd2 " << *this;
    }
    void f(T*) {
        cout << "fd3 " << *this;
    }
};

template <class T, class U>
class Specific : public Derived<T, U> {
public:
    Specific() : Derived<T, U>() { }
    Specific(T k, U v) : Derived<T, U>(k, v) { }
    void f(Base<T, U> e1, Base<T, U> e2) {
        Base<T, U>::Update(e1, e2);
        cout << "fs1 " << *this;
    }
    void f(Base<T, U> e1, Derived<T, U>& e2) {
        Base<T, U>::Update(e1, e2);
        cout << "fs2 " << *this;
    }
};

int main() {
    Base<int, double> b(1, 2);
    Derived<int, double> d(1, 3);
    Specific<int, double> s(2, 3);
    Base<int, double>* pbd = &d, *pbs = &s;
    Derived<int, double>* pds = &s;

    b.f(b, b);
    b.f(d, d);
    pbd->f(b, d);
    pbd->f(d, d);
    d.f(b, d);
    d.f(d, d);
    pbs->f(b, d);
    pbs->f(d, d);
    s.f(b, d);
    s.f(d, d);
    pds->f(b, d);
    pds->f(d, d);

    Data o(1, 3);
    Data* po = &o;
    Base<Data, double> bod(o, 2);
    Derived<Data, double> dod(o, 4);
    Base<Data, double> *pbdo = &dod;
    bod.f(o);
    dod.f(po);
    pbdo->f(po);
    return 0;
}

```

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

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

class Base
{
protected:
    string s;
public:
    Base(string ss) { s = ss; }
    int c(Base* calc, int k) {
        return f(k) - calc->f(k);
    }
    virtual int f(int k) { return s.length(); }
    virtual int h() { return s[0] - 'A' + s[1] - 'A'; }
    virtual Base** g() { return nullptr; }
    virtual void print() { cout << s << endl; }
};

class Derived : public Base
{
public:
    Derived(string s) : Base(s) { }
    int f(int k) { return stoi(s); }
    int h() { return s[0] - '0' + s[1] - '0'; }
};

class Specific : public Base
{
    Base* left;
    Base* right;
public:
    Specific(Base* cl, Base* cr)
        : Base(*cl), left(cl), right(cr) { }
    int f(int k) {
        --k;
        return left->f(k) + (k ? right->f(k) : 0);
    }
    int h() { return left->h(); }
    Base** g() { return &right; }
    virtual void print() {
        if (left)
            left->print();
        if (right)
            right->print();
    }
};

class Creator
{
    Base** arCalc;
    int size;
public:
    Creator(int n) : size(n){
        arCalc = new Base*[size]{};
    }
    int inspect(string s, char c1, char c2) {
        int ind = 0;
        for (; ind < s.length() && s[ind] >= c1 && s[ind] <= c2; ind++);
        return ind;
    }
};
```

```

Base* createObject(string s) {
    int index = inspect(s, 'A', 'Z');
    if (index == 0) {
        index = inspect(s, '0', '9');
        if (index == s.length())
            return new Derived(s);
        else
            return new Specific(createObject(s.substr(index)),
                                new Derived(s.substr(0, index)));
    } else if (index == s.length())
        return new Base(s);
    else
        return new Specific(new Base(s.substr(0,index)),
                            createObject(s.substr(index)));
}

void add(Base* calc) {
    int h = calc->h() % size;
    if (arCalc[h] == nullptr)
        arCalc[h] = calc;
    else {
        Base *cc = arCalc[h], *ccc = cc, *cccc = cc, **ppc;
        while (cc && cc->c(calc, 1) > 0) {
            cccc = ccc; ccc = cc;
            cc = (ppc = cc->g()) ? *ppc : nullptr;
        }
        if (cc == ccc) {
            arCalc[h] = new Specific(calc, cc);
        } else if (cc == nullptr) {
            if (ccc == cccc) {
                arCalc[h] = new Specific(ccc, calc);
            } else {
                *cccc->g() = new Specific(ccc, calc);
            }
        } else {
            *ccc->g() = new Specific(calc, cc);
        }
    }
}

void print() {
    for (int i = 0; i < size; i++) {
        if (arCalc[i]) {
            cout << i << endl;
            arCalc[i]->print();
        }
    }
}

};

int main()
{
    int ns = 9;
    string ars[] = { "ACA", "GDB22222", "DARK02000DASA", "73210", "ADD44444",
                    "FAULT11111", "BEBA2001BORA2002", "486", "CD650MB74MIN"
    };
    Creator creator(7);
    for (int i = 0; i < ns; i++) {
        Base* calc = creator.createObject(ars[i]);
        creator.add(calc);
    }
    creator.print();
    return 0;
}

```

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

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

template <class T, int n>
class Elem {
public:
    T val, key;
    Elem<T, n>* arr[n] = {};
    Elem() { val = T(); key = T(); }
    Elem(T v, T k) { val = v; key = k; }
    void Update() { key++; }
    Elem<T, n>* GetNext(T v, int k) {
        Elem<T, n>*& ppElem = &arr[k++ % n];
        while (*ppElem && (*ppElem)->val < v) {
            ppElem = &(*ppElem)->arr[k % n];
        }
        if (!(*ppElem)) {
            (*ppElem) = new Elem<T, n>(v, 0);
        } else if ((*ppElem)->val > v) {
            Elem<T, n>* newElem = new Elem<T, n>(v, 0);
            newElem->arr[k % n] = *ppElem;
            (*ppElem) = newElem;
        }
        return *ppElem;
    }
    bool check(T v) { return this && v % val == 0; }
    void Print(T& sv, int k) {
        if (!this) return;
        sv += val;
        if (key)
            cout << val << " " << sv << " " << endl;
        arr[++k % n]->Print(sv, (k + 1) % n);
        arr[++k % n]->Print(sv, k % n);
    }
};

template <class T, int n, int m>
class Container
{
    T arr[m + 1] = {};
    Elem<T, n>* first;
public:
    Container() : first() { }
    void Print() {
        T sv = 0;
        first->Print(sv, 0);
        cout << endl;
    }
    void Prepare() {
        T i = 0, is = 1;
        arr[i] = 0;
        arr[++i] = 1;
        while (is <= m) {
            if (!arr[i]) {
                arr[i] = i;
                for (T j = is; j <= m; j += i)
                    if (!arr[j])
                        arr[j] = i;
            }
            is = ++i * i;
        }
    }
};
```

```

void Process(T x) {
    Elem<T, n>* pCurr = first;
    while (x != 1) {
        T value = arr[x] ? arr[x] : x;
        pCurr = pCurr->GetNext(value, 1);
        x /= value;
    }
    pCurr->Update();
}

void Create(T arr[], int narr) {
    Prepare();
    first = new Elem<T, n>(1, 0);
    for (int i = 0; i < narr; ++i) {
        Process(arr[i]);
    }
}

T Search(T x, T y) {
    T res = 1;
    Elem<T, n>* pX = first;
    Elem<T, n>* pY = first;
    while (x != 1 || y != 1) {
        while (pX->check(x) && pY->check(y) && pX->val == pY->val) {
            res *= pX->val;
            x /= pX->val;
            pX = pX->arr[1];
            y /= pY->val;
            pY = pY->arr[1];
        }
        while (pX->check(x) && (!pY || pX->val < pY->val)) {
            res *= pX->val;
            x /= pX->val;
            pX = pX->arr[1];
        }
        while (pY->check(y) && (!pX || pX->val > pY->val)) {
            res *= pY->val;
            y /= pY->val;
            pY = pY->arr[1];
        }
        while (pX && x % pX->val != 0 && (!pY || pX->val <= pY->val)) {
            pX = pX->arr[0];
        }
        while (pY && y % pY->val != 0 && (!pX || pX->val >= pY->val)) {
            pY = pY->arr[0];
        }
    }
    return res;
}

};

int main() {
    int n = 8;
    int arr[] = { 32, 36, 294, 100, 343, 98, 225, 81 };
    Container<int, 2, 1000> container;
    container.Create(arr, 8);
    container.Print();
    cout << container.Search(arr[0], arr[1]) << endl;
    cout << container.Search(arr[0], arr[3]) << endl;
    cout << container.Search(arr[4], arr[5]) << endl;
    cout << container.Search(arr[1], arr[2]) << endl;
    return 0;
}

```

Answers: Object oriented programming – 2023 (Ohrid)

Task1.

```
A()
A(2)
B()
A::f
A::ff() 8
B::f
B::ff() 16
A::f
B::ff() 36
A = 8 & B = 36
A()
B(B())
A::f
B::ff() 16
B::ff()c 18
A = 16
B::f
B::ff() 324
B::ff()c 326
B = 324
~B()
~A(324)
A = 8 & B = 36
~B()
~A(36)
~A(8)
```

Task2.

```
opB+(int) 19
opB+(B) 31
B::op= 4 4
Ohrid++ 2
Ohrid++ 3
op+(ul,A)ext
op+(ul,A)ext
op+(ul,A)ext
B::op= 4 4
opC+(int) 28
opC+(B) 80
opC+(B) 276
C::op= 20 256
A::op= 4
A::op= 20
A::op= 4
max(a1,a2)
A::op= 4
```

Task3.

```
consA 1
consA 3
6 consC
consB 6
1 consB
consD 1
consA 3
consA 7
consC 7
1 consB
consD 1
consA 1
consB 1
consF 1
0 dest
consA 1
consC 1
consA 3
consC 3
F::f->2
F::f->3
F::f->2
```

Task4.

```
Base 1
Base 2
Der 2
Base 2
Der 2
Base 2
Base 2
Der 2
Part 2
Bf 5
Pf 2
Df 18
Pf 2
Pf 2
Base 1
Bg->Bf 5
Base 1
Dg->Df 10
Base 1
Bg->Pf 2
Base 2
Bg->Bf 9
Base 2
Dg->Df 10
Base 2
Bg->Pf 2
```

Task5.

```
fb1 1 4
fb4 1 7
fb2 1 6
fb4 1 12
fd1 1 19
fd1 1 38
fs2 2 10
fb4 2 48
fs2 2 55
fs2 2 93
fs1 2 100
fs1 2 138
fb5 1 3 2
fd3 1 3 4
fb6 1 3 4
```

Task6.

```
2
GDB
22222
ACA
3
73210
DARKO
DASA
2000
ADD
44444
5
486
FAULT
11111
BEBA
BORA
2002
2001
CD
MB
MIN
74
650
```

Task7.

```
2 11
3 17
5 27
7 44
7 58
3 70
5 80
7 101

288
800
686
1764
```