Object oriented programming – Elektrijada 2014 (Balaton)

Task1 (16):	Task4 (16):
Task2 (16):	Task5 (18):
Task3 (16):	Task6 (18):

Task1	Task2	Task3	Task4	Task5	Task6	Sum

<u>Task1. What is the output for the following program?</u>

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

using namespace std;

static int b;
class X {
      int *pi;
public:
      int x;
      virtual int a();
      int bb();
      void operator() (X& x);
      X(){b=0; x=-2;};
      X(int i) : pi(new int(i)) {b++;}
      X(const X &x) : pi(new int(*x.pi)){b*=3;}
      X& operator= (const X&);
};

X& X::operator= (const X& x) {
      if (this != &x) {};
      return *this;
}

int X::a(){cout << "F";this->x;return bb();}
int X::bb(){cout << "1\n";return this->x;}

void X::operator() (X& x){
      this->x = x.a() * x.bb();
      cout << this->x << endl;
}

class Derived: public X{
public:
      int a();
      virtual int bb();
};

int Derived::a(){ cout << "Hungaroring\n";return bb()*10;}
int Derived::bb(){ cout << "Kimi Raikkonen\n";return this->x * 2;}

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

void g() {
      X xa=4, xb=3;
      X xc = xa;
      xa = funF(xb);
      xc = xa;
}

class Base: public X, public Derived{
public:
      int a();
      int bb();
};

int Base::a(){ cout << "20 years\n";return bb()*10;}
int Base::bb(){ cout << "Ayrton Sena\n";return this->Derived::x * 2;}
int main(int argc, char* argv[]){
      X obj, d();
      X *base = new Derived();
      Derived *derived = new Base();
      obj.a(); base->a(); obj.bb();
      base->bb(); obj(*base); (*base)(obj);
      g();
      obj(*derived);
      cout << b << endl;
      return 0;
      cout << "\nE L E K T R I J A D A\n" << endl;
}
```

Task2. What is the output for the following program?

```
#include <iostream>
#include <iomanip>
#include <math.h>
#include <stdlib.h>

#define d -0.00126448926734961868021375957764

using namespace std;
double a = 0, r = 1, grrr = 22, devider = 8;
int n = 3, h0 = 256, equalCoo = 073;
static int mX,mY,mH;
class DrawA; class DrawB; class DrawC; class DrawD;

class Draw{
public:
      Draw(bool ord);
      ~Draw(){};
      void draw(int i, Draw& some);
      virtual void mainDraw(int level){};
protected:
      void equal(){if(mX == mY) {equalCoo++;cout << mX << endl;}}
      bool mOrd;
      int mLevel;
      double mSigned;
      double elektrijada;
};
Draw::Draw(bool ord) : mOrd(ord) {};

void Draw::draw(int i, Draw &some){
      double x, y;
      if (i > 0){
            cout<<fixed<<setprecision(3)<<a<<" "<<r<<"="<<
                  ((r*cos(a)<0.00001&&r*cos(a)>-0.00001)?0:r*cos(a))
                  <<"|"<<((r*sin(a)<0.00001&&r*sin(a)>-0.00001)?0:r*sin(a))<<endl;
            if(mOrd){
                  r = sqrt((r *cos(a) + elektrijada*cos(mSigned+grrr))
                        * (r *cos(a) + elektrijada*cos(mSigned+grrr))
                        + (r*sin(a)+elektrijada*sin(mSigned+grrr))
                        * (r*sin(a)+elektrijada*sin(mSigned+grrr)));
                  r = fabs(r-1)<0.00001?1:r;
                  x = (r *sin(a)+elektrijada*sin(mSigned+grrr));
                  y = (r *cos(a) + elektrijada*cos(mSigned+grrr));
            }else{
                  r = sqrt((r*cos(a)+elektrijada*cos(mSigned))
                        * (r*cos(a)+elektrijada*cos(mSigned))
                        + (r *sin(a)+elektrijada*sin(mSigned))
                        * (r *sin(a)+elektrijada*sin(mSigned)));
                  r = fabs(r-1)<0.00001?1:r;
                  x = (r *sin(a)+elektrijada*sin(mSigned));
                  y = (r *cos(a) + elektrijada*cos(mSigned));
            }
            a = fabs(y)<0.00001 ?
                  (x>0?1:-1)*grrr/2 : (y<0?grrr/2:0) + (y<0?grrr/2:0) + atan(x/y);
            a+=2*grrr;
            a-=(2*grrr)*(int)(a/2/grrr);
            some.mainDraw(i-1);
      }
}
```

```
class DrawA : public Draw{
public:
      DrawA(bool ord) : Draw(ord){ mSigned = grrr/devider;
            elektrijada = 2*cos(3*mSigned);}
      void mainDraw(int level);
};

class DrawB : public Draw{
public:
      DrawB(bool ord) : Draw(ord){mSigned = 3*grrr/devider;
            elektrijada = 2*cos(mSigned); }
      void mainDraw(int level);
};

class DrawC : public Draw{
public:
      DrawC(bool ord) : Draw(ord){mSigned = 5*grrr/devider;
            elektrijada = 2*cos(3*mSigned/5); }
      void mainDraw(int level);
};

class DrawD : public Draw{
public:
      DrawD(bool ord) : Draw(ord){mSigned = 7 * grrr/devider;
            elektrijada = 2*cos(3*(grrr-mSigned));}
      void mainDraw(int level);
};

void DrawA::mainDraw(int level){if((level+3)&010){DrawB b(false);draw(level, b);}
      else{DrawB c(true);draw(level, c);}}
void DrawB::mainDraw(int level){if((level+3)&010){DrawC c(false);draw(level, c);}
      else{DrawC f(true);draw(level, f);}}
void DrawC::mainDraw(int level){if((level+3)&010){DrawD a(false);draw(level, a);}
      else{DrawD b(true);draw(level, b);}}
void DrawD::mainDraw(int level){if((level+3)&010){DrawA a(true);draw(level, a);}
      else{DrawA b(true);draw(level, b);}}

int main(int argc, char* argv[]){

      int step = 0, hlen = h0, x0 = hlen / 2, y0 = x0, i = 0;
      i +=011;
      grrr = grrr/(i-2)+d;
      a = grrr-grrr/2+grrr;
      DrawA A(false);
      A.mainDraw(--i);
      cout << equalCoo << endl;
      return 0;
}
```

Task3. What is the output for the following program?

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

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

Task4. What is the output for the following program?

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

int data = 0;
class A {
public:
      A( ){data++;}
      A& operator++( ) {cout << "1 " << data-- << endl; A* b = new A; return *b;}
      int operator++(int) {cout << "2 " << data++ << endl; A* b = new A; return --
data;}
      int operator--( ) {cout << "3 " << ++data << endl; A* b = new A; return data++;}
      A& operator--(int) {cout << "4 " << --data << endl; A* b = new A; return *b;}
      int operator+(A& b) { data++; return data;}
      A& operator-(A& b) { data++; return ++b;}
      ~A( ){ }
};

int operator+(A a, int k) {cout << "operator+(int,A)ext" << endl; return a+a ;}
A& operator+ (int k, A a) {cout << "operator+(A,int)ext" << endl; return a-a; }
A& operator-(int a, A k) {cout << "operator-(int,A)ext" << endl ; return k-k; }
int operator- (A k, int a) {cout << "operator-(A,int)ext" << endl; return data+1; }

int main()
{
      A a, b;

      ++a---b+++b---++a;
      ++a---b+(++b-(--++a));

      return (int)0;
}
```

Task5. What is the output for the following program?

```
#include "iostream"
using namespace std;

class Item {
public:
      int simbol;
      Item() { simbol = -1; };
      Item(int s) { simbol = s; };
      virtual int GetType() = 0;
      virtual Item* Process(Item& it, Item* pItem = NULL) = 0;
      virtual void Print() { cout << simbol << " "; };
};

class Type : public Item {
public:
      Type(int t) : Item(t) {};
      int GetType() {
            int type = 0;
            if (simbol >= 0) for (int t=simbol; t; t>>=1) type++;
            else type = simbol;
            return type+1;
      };
      Item* Process(Item& it, Item* pItem = NULL) { return this; };
};

class Value : public Item {
public:
      Value(int v) : Item(v) {};
      int GetType() { return simbol / 0x10; };
      Item* Process(Item& it, Item* pItem) { return this; };
};

class Creator;
class Data : public Item {
protected:
      Item *pCurr, *pPrev;
      Creator *pCreate;
      friend class Creator;
public:
      Data() : Item() {
            pCurr = pPrev = NULL;
            pCreate = NULL;
      };
      Data(Creator *pCrt, Item* pC, Item* pP, int s)
            : pCurr(pC), pPrev(pP), pCreate(pCrt), Item(s) { };
      int GetType() {
            int type = 0;
            for (int t=simbol; t; t>>=1) type++;
            return type+1;
      };
      Item* Process(Item& it, Item* pItem = NULL);
      void Print() {
            if (pPrev) pPrev->Print();
            cout << simbol << " ";
      };
};

class DataValue : public Data {
public:
      DataValue(Creator* pCr, Item* p, int s) : Data(pCr, NULL, p, s) { };
};
```

```
class DataProcess : public Data {
public:
      DataProcess(Creator* pCr, Item* pC, Item* pP, int s) : Data(pCr, pC, pP, s) { };
};

class Creator {
      Item& makeItem(int v) {
            Item *rItem = NULL;
            if (v / 0x10) rItem = new Value(v);
            else rItem = new Type(v);
            return *rItem;
      };
public:
      Data* makeData(Data* pData, int s, int type) {
            Item* pPrev = NULL;
            if (pData != NULL) pPrev = pData->pPrev;
            if (type == 0) return new DataValue(this, pPrev, s);
            else  return new DataProcess(this, pData, pPrev, s);
      };
      Item* makeData(Data* pData, int v) {
            Item& it = makeItem(v);
            if (pData == NULL) return
                  new DataProcess(this, NULL, new DataValue(this, NULL, v), -1);
            else
                  if (v / 0x10) return pData->Process(it);
                  else return pData->Process(it, pData->pPrev);
      };
};

Item* Data::Process(Item& it, Item* pItem) {
      if (simbol == -1) {
            simbol = it.simbol;
            return this;
      } else if (pItem == NULL) {
            pPrev = pCreate->makeData(this, it.simbol, 0);
            return this;
      } else {
            this->pPrev = pItem;
            if (it.GetType() < GetType()) {
                  Item* pTmp = pCurr;
                  pPrev = pItem = pCreate->makeData(this, simbol, 0);
                  if (pTmp == NULL) {
                        simbol = it.simbol;
                        return this;
                  } else return pTmp->Process(it, pItem);
            } else if (it.GetType() == GetType()) {
                  pPrev = pItem = pCreate->makeData(this, simbol, 0);
                  simbol = it.simbol;
                  return this;
            } else
                  return pCreate->makeData(this, it.simbol, 1);
      }
}

int main() {
      char arr[] = {16,2,17,4,18,5,16,3,19,5,17,4,20,8,21,5,22,-1};
      Data *pData = NULL;
      Creator create;
      for (int i=0; i<18; i++)
            pData = (Data *) create.makeData(pData, arr[i]);
      pData->Print();
      return 0;
}
```

Task6. What is the output for the following program?

```
#include "iostream"
using namespace std;

template<class Tip, int n>
class Data {
	Tip arVal[n];
public:
	Data() { for (int i=0; i<n; i++) arVal[i] = -1;	};
	Data(Tip arV[]) { for (int i=0; i<n; i++) arVal[i] = arV[i]; };
	Tip Value() {
		Tip val = 0;
		for (int i=0; i<n/2; i++)
			val += (arVal[i+n/2]-arVal[i]);
		return val;
	};
	void Print() {
		for (int i=0; i<n; i++) cout << arVal[i] << " ";
		cout << endl;
	};
	void Combine(Data& data) {
		for (int i=0; i<n/2; i++)
			if (arVal[i] == -1) {
				arVal[i] = data.arVal[i];
				arVal[i+n/2] = data.arVal[i+n/2];
			} else {
				arVal[i] = arVal[i]-data.arVal[i]>0 ? data.arVal[i] : arVal[i];
				arVal[i+n/2] = arVal[i+n/2]-data.arVal[i+n/2]<0
					? data.arVal[i+n/2] : arVal[i+n/2];
			}
	};
};

template<class Tip, int dim>
class Set {
	Tip arData[dim];
	Set* arSet[dim+1];
	int n;
	int Overlap(Tip data) {
		arData[n++] = data;
		int ii=1, dd = 0, norm = 10000;
		for (int i=0; i<n-1; i++, ii<<=1);
		for (int d=1, i=0; i<n-1; d<<=1, i++) {
			dd = Compare(d|ii, norm) ? d|ii : dd;
		}
		n--;
		for (dd=dd^ii, ii=0; dd>1; ii++, dd>>=1);
		return ii;
	};
	Set* Duplicate() {
		int dd = 0, norm = 10000;
		for (int i=1, s=0; s<dim-1; i++) {
			s = 0;
			for (int d = i; d; s+=d%2, d>>=1);
			if (s == dim/2)
				dd = Compare(i, norm) ? i : dd;
		}
		return Reorder(dd);
	};
	bool Compare(int d, int& norm) {
		Tip data[2];
		for (int i=0; i<n; d>>=1, i++)
			data[d%2].Combine(arData[i]);
		if (data[0].Value() + data[1].Value() < norm) {
			norm = data[0].Value() + data[1].Value();
			return true;
		}
		return false;
	};
```

```
        Set* Reorder(int dd) {
                Set *pSet = new Set();
                for (int i=0, ii=0; i<n; i++, dd>>=1) {
                        if (dd%2) {
                                if (arSet[i]) arSet[i]->arSet[dim] = pSet;
                                pSet->arData[pSet->n] = arData[i];
                                pSet->arSet[pSet->n++] = arSet[i];
                        } else {
                                arData[ii] = arData[i];
                                arSet[ii++] = arSet[i];
                        }
                }
                for (int j=n-=dim/2; j<dim; j++) arSet[j] = NULL;
                if (!arSet[dim]) {
                        Set *pSetP = new Set();
                        pSetP->arSet[pSetP->n++] = this;
                        arSet[dim] = pSetP;
                }
                pSet->arSet[dim] = arSet[dim];
                return arSet[dim]->Update(this, pSet);
        };
        Set* Update(Set* pOld, Set* pNew) {
                int i = 0;
                for ( ; i<n && arSet[i]!=pOld; i++);
                arData[i] = pOld->Size();
                arData[n] = pNew->Size();
                arSet[n++] = pNew;
                if (n >= dim) return Duplicate();
                return this;
        };
        Tip Size() {
                Tip data;
                for (int i=0; i<n; i++) data.Combine(arData[i]);
                return data;
        };
public:
        Set() : n(0) {
                for (int i=0; i<dim+1; i++) arSet[i] = NULL;
        };
        void Print() {
                if (arSet[n-1]) {
                        for (int i=0; i<n; i++) arSet[i]->Print();
                } else {
                        for (int i=0; i<n; i++) arData[i].Print();
                        cout << endl;
                }
        };
        Set* Update(Tip& data) {
                if (n>0 && arSet[n-1]) {
                        int ind = Overlap(data);
                        Set *pSet = arSet[ind]->Update(data);
                        return pSet != arSet[ind] ? pSet : this;
                } else {
                        arData[n++] = data;
                        Size();
                        return n >= dim ? Duplicate() : this;
                }
        };
};

int main() {
        int arVal[12][4] = { {1,5,3,7}, {4,3,6,7}, {3,4,7,6}, {5,7,8,9}, {8,4,10,7}, {6,3,7,5},
                {13,11,14,15}, {2,12,5,14}, {5,13,7,15}, {11,6,13,8}, {12,1,15,4}, {9,10,11,12} };
        Set< Data<int, 4>, 5 > *pSet = new Set< Data<int, 4>, 5 >;
        for (int i=0; i<12; i++) {
                Data<int, 4> data(arVal[i]);
                pSet = pSet->Update(data);
        }
        pSet->Print();
        return 0;
}
```

Solutions: Object oriented programming – Balaton 2014

Task1.

```
F1
Hungaroring
Kimi Raikkonen
1
1
Hungaroring
Kimi Raikkonen
1
80
F1
1
6400
20 years
Ayrton Sena
1
80
57
```

Task2.

```
4.712 1.000=0.000|-1.000
5.498 1.000=0.707|-0.707
0.000 1.000=1.000|0.000
0.785 1.000=0.707|0.707
1.571 1.000=0.000|1.000
2.356 1.000=-0.707|0.707
3.142 1.000=-1.000|0.000
3.927 1.000=-0.707|-0.707
59
```

Task3.

```
1 OOP
2 OOP
1 JADA
2 OOP
-8 OOP
-3 JADA
6 OOP
13 OOP
-52 OOP
-12 JADA
6 JADA
-25 JADA
-2 JADA
-51 OOP
2 JADA
0 JADA
```

Task4.

```
1 2
4 1
2 2
4 2
1 3
operator-(A,int)ext
operator+(A,int)ext
1 4
1 5
1 5
3 6
1 8
operator-(A,int)ext
4 7
1 8
1 9
operator+(int,A)ext
```

Task5.

```
16 17 18 4 16 5 2 19 17 5 20 21 8 4 22 5
3 -1
```

Task6.

```
13 11 14 15
11 6 13 8
12 1 15 4

5 7 8 9
8 4 10 7

2 12 5 14
5 13 7 15
9 10 11 12

1 5 3 7
4 3 6 7

3 4 7 6
6 3 7 5
```