Object oriented programming – Elektrijada 2015 (Bečići)

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;


int id = 1;


class ClassA {
public:
      ClassA() { id = 1;}
      ClassA(int objectCode) {
            objectID = 0;
            objectVAL = objectCode;
            cout << objectID << " " << objectVAL << " constructor" << endl;
      }
      ClassA(int objectNumber, int objectCode)
                  : objectID(objectNumber), objectVAL(objectCode) {
            cout << objectID << " " << objectVAL << " constructor" << endl;
      }
      ClassA Print() {
            cout << objectID << " " << objectVAL << endl;
            return *this;
      }
      int getID() { return objectID; }
      int getVAL() { return objectVAL; }
      virtual ClassA funf(int i) {
            ClassA af(*this);
            af.fun(i);
            af.fun(*this);
            return af.Print();
      }
      void fun(int i) {
            objectID *= i;
      }
      virtual void fun(ClassA aa) {
            objectID += aa.objectID;
            objectVAL += aa.objectVAL;
      }
protected:
      int objectID;
      int objectVAL;
};



class ClassB : public ClassA {
public:
      ClassB(): ClassA(id) {ClassA a(id);}
      ClassB(ClassA a): ClassA(a) {ClassA aa(id);}
      ClassA funf(int i) {
            ClassA af(*this);
            af.fun(-i);
            af.fun(*this);
            return af.Print();
      }
      void fun(ClassA aa) {
            objectID -= aa.getID();
            objectVAL -= aa.getVAL();
      }
};
```

```
class ClassC : public ClassA {
protected:
      ClassA *a;
public:
      ClassC(int i) : ClassA(i)  { a=new ClassA(i); }
      ClassC(ClassA aa) : ClassA(aa) { a=new ClassA(aa); }
      ClassA funf(int i) {
            ClassA af(a->funf(i*i));
            af.fun(i);
            af.fun(*this);
            return af.Print();
      }
};


class ClassD : public ClassC {
public:
      ClassD(int i) : ClassC(i)  { a=new ClassB(i); }
      ClassD(ClassA aa) : ClassC(aa) { a=new ClassB(aa); }
      ClassA funf(int i) {
            ClassB af(a->funf(i*i));
            af.fun(-i);
            af.fun(*this);
            return af.Print();
      }
};


int main() {
      ClassA a(id, id+1);
      a.fun(id+1);
      a.funf(id*=2);
      ClassB b(a);
      b.fun(id);
      ClassB bb(id+1);
      bb.funf(id*=2);
      ClassC c(bb);
      c.funf(id);
      ClassD d(id*=2);
      d.funf(id/2);
      ClassD dd(c);
      dd.funf(id);
      return 0;

}
```

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

```
#include <iostream>

using namespace std;
int c = 0;
class Elektrijada  {
public:
      Elektrijada(){ c++; }
};
class Becici : public Elektrijada {
public:
      Becici(){ c = c << 1; }
};
class OOP : public Becici {
public:
      OOP(){ c << 1; }
};
void f(int i) {
      switch (i) {
      case 1: throw 1.2f; break;
      case 2: throw Elektrijada(); break;
      case 3: throw 3.2; break;
      case 4: throw Becici(); break;
      case 5: throw new OOP(); break;
      }
}
void g(int i) {
      switch (i) {
      case 0: throw Elektrijada(); break;
      case 6: throw 2.3; break;
      case 7: throw 3.2f; break;
      case 8: throw OOP(); break;
      case 9: throw new Becici(); break;
      }
}

int main() {
      for (int i = 0; i<10; i++) {
            try {
                  f(i);
                  try {
                        g(i);
                  }
                  catch (Becici &becici) { cout << "Becici"; }
                  catch (OOP &my) { cout << "OOP"; }
                  catch (Elektrijada &my) { cout << "Elektrijada"; }
                  catch (double) { cout << "double"; }
                  catch (float) { cout << "float"; }
                  catch (OOP *my) { cout << "*00P*"; }
            }
            catch (Elektrijada *my) { cout << "*Elektrij@da*"; }
            catch (Becici *becici) { cout << "*Becici*"; }
            catch (float) { cout << "flo@t"; }
            catch (double) { cout << "d0uble"; }
            catch (OOP &my) { cout << "00P"; }
            catch (Elektrijada &my) { cout << "Elektrij@da"; }
            catch (Becici &becici) { cout << "Becici"; }
            cout << i << endl;
      }
      cout << c;
      return 0;
}
```

<u>Task 3. What is the output for the following program?</u>

```
#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;
            id *= -1;
      }
      ~A() {
            id *= -1;
            id > 0 ? (cout << objID+id << " JADA " << endl, id++)
                        : (cout << objID-id << " JADA " << endl, id--);
      }
private:
      int objID;
};
class B : virtual public A {
      A a;
public:
      B() : A(id) { }
      B(int i) : A(i) { static A b(i); }
      ~B() { A b(); }
};
class C : public A {
public:
      C(int i) : A(id) {
            //on create :-)
            A b(i);
      }
      virtual ~C() {
            //on remove :-\
            A b(1);
      }
};
class D : public B, virtual public C, virtual public A {
public:
      D(int i) : B(i), A(i), C(1) { static A a(i); }
};
int main() {
      C c(), c();
      D b(id);
      c(), c();
      exit(0);
}
B b(id++);
C c() {
      static C c(id++);
      return c;
};
C c();
```

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

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

#define dim 20
int b = 0;

class Elem {
protected:
      int data;
      int weight;
public:
      Elem() { data = 0; weight = 1; b+=2; }
      Elem(int d, int w) : data(d), weight(w) {}
      Elem(Elem& el) : data(el.data), weight(el.weight) { b++; }
      Elem& operator=(const Elem &el) {
            data = el.data;
            weight = el.weight;
            b+=3;
            return *this;
      }
      bool operator<(Elem &el) {
            return Value() < el.Value();
      }
      bool Compare(Elem *el) {
            return Size() < el->Size();
      }
      virtual int Size() {
            return data * weight;
      }
      virtual int Value() {
            return data;
      }
      virtual void Print() {
            cout << data << " " <<  weight << endl;
      }
};

class Container : public Elem
{
      Elem *els[dim];
      int n;
public:
      Container() : n(0){
            for (int i=0; i<dim; i++)
                  els[i] = NULL;
      }
      void Add(Elem *elem) {
            els[n++] = elem;
            data += elem->Value();
      }
      int Size() {
            int res = 0;
            for (int i=0; i<n; i++)
                  res += els[i]->Size();
            return res;
      }
      int Value() {
            int res = 0;
            for (int i=0; i<n; i++)
                  res += els[i]->Value();
            return res;
      }
```

```
	void Rearrange() {
		for (int i=1; i<n; i++) {
			int j = i;
			Elem* el = els[i];
			for (; j>0 && els[j] < els[j-1]; j--)
				els[j] = els[j-1];
			els[j] = el;
		}
	}
	Container* Join(Container &con) {
		Container* res = new Container();
		for (int i=0, ii=0; i<n || ii<con.n; ) {
			if (i<n && ii<con.n) {
				if (els[i]->Compare(con.els[ii]))
					res->Add(els[i++]);
				else
					res->Add(con.els[ii++]);
			} else if (i<n) {
				res->Add(els[i++]);
			} else {
				res->Add(con.els[ii++]);
			}
		}
		return res;
	}
	void Print() {
		Elem::Print();
		for (int i=0; i<n; i++)
			els[i]->Print();
	}
};

int main() {
	const int size = 5;
	int data = 64, weigth = 2;
	Elem arr[15] = { };
	Container con1, con2, *con3;
	for (int i=0; i<size; i++) {
		arr[2*i] = Elem(data-=data/2, weigth*=2);
		con1.Add(&arr[2*i]);
		arr[2*i+1] = Elem(data%size, weigth/4);
		con2.Add(&arr[2*i+1]);
	}
	con1.Rearrange();
	con1.Print();
	cout << b << endl;
	con2.Rearrange();
	Container con4 = con2;
	con4.Print();
	cout << b << endl;
	con3 = con1.Join(con2);
	con3->Print();
	cout << b << endl;
	return 0;
}
```

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

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

using namespace std;

class Derived
{
public:
      Derived(int broj);
      virtual void TillTheEnd();
protected:
      virtual Derived* Create(int broj);
      Derived* left;
      Derived* right;
      int info;
};


class Option : public Derived
{
public:
      Option(int broj);
      virtual void TillTheEnd();
protected:
      virtual Derived* Create(int broj);
};


class Sun : public Derived
{
public:
      Sun(int broj);
      virtual void TillTheEnd();
protected:
      virtual Derived* Create(int broj);
};


Derived::Derived(int broj)
{
      left = right = NULL;
      info = broj;  if (broj>0)   Create(broj >> 3);
}

Derived* Derived::Create(int broj)
{
      if (info & 1) right = new Option(info >> 1);
      else   left = new Sun(info >> 2);
      return this;
}

void Derived::TillTheEnd()
{
      cout << "Derived: " << info<<endl;
      if (left) left->TillTheEnd();
      if (right) right->TillTheEnd();
}

Option::Option(int broj) : Derived(broj) {}
```

```
Derived* Option::Create(int broj)
{
      if (info & 1 ^ 1)     right = new Sun(info >> 1);
      else   left = new Derived(info >> 2);
      return left;
}

void Option::TillTheEnd()
{
      if (left)     left->TillTheEnd();
      cout << "Option: " << info << endl;
      if (right)    right->TillTheEnd();
}

Sun::Sun(int broj) : Derived(broj) {}

Derived* Sun::Create(int broj)
{
      if (info & 1 ^ 1)     right = new Derived(info >> 1);
      else   left = new Option(info >> 2);
      return left;
}

void Sun::TillTheEnd()
{
      if (left)     left->TillTheEnd();
      if (right)    right->TillTheEnd();
      cout << "Sun: " << info << endl;
}

int main(){
      int broj = 0x6C;
      Derived a(broj);
      a.TillTheEnd();
      return 0;
}
```

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

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

template <class T1, class T2>
class Node {
public:
    T1 value;
    T2 data;
    Node<T1, T2> *prev, *par, *next;
    Node(T1 v, Node *p, Node *pr, Node *n, T2 d=T2())
        : prev(p), par(pr), next(n), value(v), data(d) { }
    void Update() {
        if (prev != NULL) prev->next = next;
        if (next != NULL) next->prev = prev;
    }
    Node<T1, T2>* Update(T2 newData) {
        Node<T1, T2> *ptr2 = NULL;
        if (prev != NULL || ptr2->prev->data > newData) {
            Update();
            for (ptr2=this; ptr2->prev!=NULL && ptr2->prev->data>newData; ptr2=ptr2->prev);
            if (ptr2->prev != NULL) ptr2->prev->next = this;
            prev = ptr2->prev;
            if (ptr2 != NULL) ptr2->prev = this;
            next = ptr2;
        }
        data = newData;
        return this;
    }
};

template <class T1, class T2>
class Content {
    Node<T1, T2> *first;
public:
    Content() : first(NULL) { }
    bool Empty() { return first == NULL; }
    void Add(Node<T1, T2> *node, T2 data) {
        Node<T1, T2> *ptr = first;
        if (ptr != NULL && ptr->data < data) {
            for ( ; ptr->next != NULL && ptr->next->data < data; ptr = ptr->next);
            Node<T1, T2> *ptrNew = new Node<T1, T2>(0, ptr, node, ptr->next, data);
            if (ptr->next != NULL) ptr->next->prev = ptrNew;
            ptr->next = ptrNew;
            node->prev = ptrNew;
        } else {
            first = new Node<T1, T2>(0, NULL, node, first, data);
            node->prev = first;
        }
    }
    Node<T1, T2>* Get() {
        Node<T1, T2> *ptr = first;
        first = first->next;
        return ptr;
    }
    bool Update(Node<T1, T2>* ptr) {
        if (ptr->next == first) first = ptr;
        return first == ptr;
    }
    Node<T1, T2>* Set(Node<T1, T2>* ptr) {
        ptr->next = first;
        ptr->prev = NULL;
        if (first) first->prev = ptr;
        return first = ptr;
    }
    void Print() {
        for (Node<T1, T2> *ptr = first; ptr != NULL; ptr = ptr->next)
            cout << ptr->data << " - " << ptr->par->value << endl;
        cout << endl;
    }
};
```

```
template <class T1, class T2>
class Collection {
    Node<T1, T2> *base;
    void Change(Content<T1, T2> &conn1, Content<T1, T2> &conn2, Node<T1, T2> *ptr1,
                           T2 data, bool bFirst) {
        for (Node<T1, T2> *ptr2 = ptr1->par; ptr2 != NULL; ptr2 = ptr2->par) {
            int newData = data + ptr2->data + ptr2->next->data;
            if (bFirst && ptr2->next->prev == NULL) {
                conn1.Add(ptr2->next, newData);
            } else if (ptr2->next->prev->data > newData) {
                Node<T1, T2>* res = ptr2->next->prev->Update(newData);
                if (!conn1.Update(res))
                    conn2.Update(res);
                Change(conn1, conn2, ptr2->next, ptr2->next->prev->data-ptr2->next->data, false);
            }
        }
    }
public:
    Collection() : base(NULL) { }
    void Read(T1 *ar1, T2 *ar2, int n) {
        Node<T1, T2> **arCorr = new Node<T1, T2>*[n];
        for (int i=0; i<n; i++)
            arCorr[i] = NULL;
        for (int i=0; i<n; i+=3) {
            if (arCorr[ar1[i]] == NULL)
                arCorr[ar1[i]] = new Node<T1, T2>(ar1[i], NULL, NULL, NULL);
            if (arCorr[ar1[i+1]] == NULL)
                arCorr[ar1[i+1]] = new Node<T1, T2>(ar1[i+1], NULL, NULL, NULL);
            arCorr[ar1[i]]->par = new Node<T1, T2>(0, NULL, arCorr[ar1[i]]->par,
                                                   arCorr[ar1[i+1]], ar1[i+2]);
        }
        for (int i=1; i<n && arCorr[i] != NULL; i++) {
            arCorr[i]->next = arCorr[i+1];
            arCorr[i]->data = ar2[i-1];
        }
        base = arCorr[1];
        delete[] arCorr;
    }
    void Print() {
        for (Node<T1, T2> *ptr1 = base; ptr1 != NULL; ptr1 = ptr1->next, cout << endl) {
            cout << ptr1->value << " + " << ptr1->data;
            for (Node<T1, T2> *ptr2 = ptr1->par; ptr2 != NULL; ptr2 = ptr2->par)
                cout << " | " << ptr2->next->value << " " << ptr2->data;
        }
        cout << endl;
    }
    void Process(T1 val1, T1 val2) {
        Content<T1, T2> conn1, conn2;
        Node<T1, T2> *ptr1 = base;
        for ( ; ptr1 != NULL; ptr1 = ptr1->next)
            ptr1->prev = NULL;
        for (ptr1=base; ptr1->value != val1; ptr1 = ptr1->next);
        conn1.Add(ptr1, ptr1->data);
        while (!conn1.Empty() && ptr1->par->value != val2) {
            ptr1 = conn2.Set(conn1.Get());
            Change(conn1, conn2, ptr1->par, ptr1->data-ptr1->par->data, true);
        }
        conn1.Print();
        conn2.Print();
    }
};

int main() {
    const int num = 42;
    int arr1[] = { 2,4,8, 1,2,6, 5,3,8, 3,6,5, 4,3,5, 5,1,3, 1,4,7, 2,5,4, 6,1,5,
        4,4,9, 3,2,2, 5,6,1, 2,3,5, 6,2,3 };
    int arr2[] = { 0, 4, 7, 9, 2, 3, 6, 8, 5 };
    Collection<int, int> coll;
    coll.Read(arr1, arr2, num);
    coll.Print();
    coll.Process(3, 1);
    return 0;
}
```

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

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


template <class T, int n>
class Data {
public:
       T arVal[n];
       Data() {
              for (int i=0; i<n; i++) arVal[i] = T();
       }
       Data(T arValues[]) {
              for (int i=0; i<n; i++) arVal[i] = arValues[i];
       }
       int CompareAndCorrect(Data<T, n> &data, T corr) {
              int res = 0;
              for (int i=n-1, p=1; i>=0; i--, p*=2)
                     if (arVal[i] >= data.arVal[i]) {
                            res += p;
                            data.arVal[i] += corr / 2;
                     } else {
                            data.arVal[i] -= corr / 2;
                     }
              return res;
       }
       void Print() {
              for (int i=0; i<n; i++) cout << arVal[i] << " ";
       }
};


template <class T, int n>
class Element {
public:
       Data<T, n> *data;
       T val;
       Element<T, n> *ptr;
       Element() {
              data = NULL; val = T(); ptr = NULL;
       }
       Element(Data<T, n> *d, T v, Element<T, n>* p) {
              data = d; val = v;  ptr = p;
       }
       T GetValue() {
              if (ptr != NULL) return val + ptr->GetValue();
              else return val;
       }
       void SetValue(T value) { val = value; }
       Data<T, n>* Create(T corr) {
              T arV[n];
              for (int i=0; i<n; i++)
                     arV[i] = (2 * corr) * (data->arVal[i] / (2 * corr)) + corr;
              return new Data<T, n>(arV);
       }
       void Print(bool bVal = true) {
              if (bVal) cout << val << " | ";
              if (data != NULL) data->Print();
              if (ptr != NULL) {
                     cout << "| ";
                     ptr->Print(false);
              }
              if (bVal) cout << endl;
       }
};
```

```
template <class T, int n, int m>
class Collection {
	Element<T, n> **arElem;
	Data<T, n> data;
	T corr;
	int fact;
	void Connect(int num, T valOld, T valNew, T corr) {
		int numNew = (num - 1) / fact;
		T vOld = T(), vNew = T();
		if (arElem[numNew] != NULL) {
			vOld = arElem[numNew]->GetValue();
			vNew = vOld-valOld+valNew;
			arElem[numNew]->SetValue(vNew);
		} else {
			Data<T, n> *data = arElem[num]->Create(corr);
			vNew = valNew;
			arElem[numNew] = new Element<T, n>(data, valNew, arElem[numNew]);
		}
		if (numNew != 0)
			Connect(numNew, vOld, vNew, 2*corr);
	}
	int Find(Data<T, n> dt, Data<T, n> data, T &corr, int mm, int num) {
		if (mm == 1) return num;
		num *= fact;
		num += dt.CompareAndCorrect(data, corr) + 1;
		corr /= 2;
		return Find(dt, data, corr, mm-1, num);
	}
	void Search(Data<T, n> dt, T corr, int num) {
		if (num < m*m*m && arElem[num] != NULL) {
			arElem[num]->Print();
			int numNew = dt.CompareAndCorrect(*arElem[num]->data, corr);
			for (int i=0; i<fact; i++) {
				int numTmp = (~i & numNew) | (i ^ numNew);
				Search(dt, corr/2, fact*num+numTmp+1);
			}
		}
	}
public:
	Collection(Data<T, n> d) : fact(n*n), data(d) {
		arElem = new Element<T, n>*[m*m*m];
		for (int i=0; i<m*m*m; i++) arElem[i] = NULL;
		corr = data.arVal[1];
	}
	void Search(Data<T, n> dt) {  Search(dt, corr, 0); }
	void Create(Data<T, n> *arD, int sz) {
		for (int i=0; i<sz; i++) {
			T valOld = T(), cr = corr;
			int num = Find(arD[i], data, cr, m, 0);
			if (arElem[num] != NULL)
				valOld = arElem[num]->GetValue();
			arElem[num] = new Element<T, n>(&arD[i], T(1), arElem[num]);
			T valNew = arElem[num]->GetValue();
			Connect(num, valOld, valNew, 2*cr);
		}
	}
};


int main() {
	int size = 16;
	int data[] = { 24,8, 17,1, 31,3, 26,2, 19,11, 22,6, 17,13, 27,13,
		29,5, 23,14, 18,3, 30,9, 23,3, 25,15, 31,14, 22,10 };
	Data<int, 2> arData[15], collData(data);
	for (int i=1; i<size; i++)
		arData[i-1] = Data<int,2>(&data[2*i]);
	Collection<int, 2, 3> coll(collData);
	coll.Create(arData, size-2);
	coll.Search(arData[size-2]);
	return 0;
}
```

Solutions: Object oriented programming – Bečići 2015

Task1.

```
1 2 constructor
6 4
0 2 constructor
0 2 constructor
0 3 constructor
0 2 constructor
0 6
0 6
0 9
0 8 constructor
0 8 constructor
0 8 constructor
0 8 constructor
0 16
0 8 constructor
0 -4 constructor
0 12
0 8 constructor
0 6
0 8 constructor
0 -8 constructor
0 11
```

Task2.

```
Elektrijada0
flo@t1
Elektrij@da2
d0uble3
Elektrij@da4
*Elektrij@da*5
double6
float7
Becici8
*Elektrij@da*9
62
```

Task3.

```
2 OOP
5 OOP
-9 OOP
19 OOP
-17 OOP
17 JADA
32 OOP
-61 OOP
30 OOP
-2 JADA
-1 JADA
0 JADA
1 JADA
29 JADA
37 JADA
36 JADA
37 JADA
```

Task4.

```
62 1
32 4
16 8
8 16
4 32
2 64
64
12 1
2 1
1 2
3 4
4 8
2 16
65
74 1
2 1
1 2
3 4
4 8
2 16
32 4
16 8
8 16
4 32
2 64
67
```

Task5.

```
Derived: 108
Option: 13
Option: 0
Sun: 1
Option: 6
Sun: 27
```

Task6.

```
1 + 0 | 4 7 | 2 6
2 + 4 | 3 5 | 5 4 | 4 8
3 + 7 | 2 2 | 6 5
4 + 9 | 4 9 | 3 5
5 + 2 | 6 1 | 1 3 | 3 8
6 + 3 | 2 3 | 1 5

19 - 4

9 - 1
8 - 6
8 - 5
6 - 2
7 - 3
```

Task7.

```
14 | 24 8
3 | 20 12
1 | 23 14
1 | 19 11
1 | 17 13
4 | 20 4
1 | 22 6
1 | 23 3
1 | 18 3 | 17 1
4 | 28 12
1 | 25 15 | 27 13
1 | 30 9
1 | 31 14
3 | 28 4
1 | 26 2
1 | 29 5
1 | 31 3
```