Object oriented programming – Elektrijada 2019 (Sunny Beach)

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 2019 (Sunny Beach)

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

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

int g = 2019;
int x = 59;
static int id = 1;

class Competition {
protected:
      int year;
public:
      Competition() : year(g) { cout << "Competition " << year << endl; }
      Competition(int i) { year = i; cout << "Competition " << i << endl; }
      Competition(Competition& e) : year(g) { cout << "Competition " << e.year << endl;
}
      Competition(const Competition& e) : year(g) {
            cout << "Competition copy from " << e.year << endl; }
      ~Competition() { cout << "See you next year!" << endl; }
};

const Competition elektrijada(2018);

class OOP : public Competition {
public:
      int year;
      OOP() { cout << "OOP " << year << endl; }
      OOP(int i) : year(g), Competition(i) { cout << "OOP " << year << endl; }
      OOP(const OOP& d) : year(g) { cout << "OOP copy from " << year << endl; }
};

class Student {
private:
      char initial = 'X';
public:
      Student() { cout << id++ << endl; }
      Student(char i) : initial(i) { cout << id++ << initial << endl; }
      Student(const Student& s) { cout << id++ << s.initial << endl; }
      Student& operator = (const Student& s) {
            initial = s.initial;
            cout << --id << s.initial << endl;
            return *this;
      }
      ~Student() { cout << initial << " goes home :(" << endl; }

      void registerFor(Competition c) {
            cout << initial << " registered for competition!" << endl;
      }

      void registerFor(OOP c) {
            cout << initial << " registered for OOP!" << endl;
      }

      void compete() & { std::cout << initial << " won!" << std::endl; }
      void compete() && { std::cout << initial << " lost!" << std::endl; }
};

void registerStudent(Student c)
{
      Student s(c);
      s.registerFor(elektrijada);
}
```

```
int main()
{
      Student a('A');
      registerStudent('B');
      static const OOP oop(2020);
      a.compete();
      a = 'C';
      a.registerFor(oop);
      Student('D').compete();
      exit(0);
}
```

Task 2. 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;
		cout << "cons A (" << id << ")" << endl;
	}
	~A() {
		id *= -1;
		objID = id > 0 ? -objID : ++objID;
		cout << objID << " dest " << endl;
	}
	virtual bool f(int objNum) {
		bool val = objID < objNum;
		cout << "A::f -> " << val << endl;
		return val;
	}
private:
	int objID;
};
class B : public A {
	A a;
public:
	B() : A(id) { cout << "cons B" << endl; }
	B(int i) : A(i) { A a(i); }
	//nonvirtual destructor :-/
	~B() { A b(int a = 5); }
	bool f(int objNum) { return A::f(objNum) || a.f(objNum); }
};
class C : virtual public A {
public:
	C(int i) : A(id) {
		cout << "cons C" << endl;
		A b(i);
	}
	//virtual destructor :-\
	virtual ~C() { B b(1); }
};
class D : public B, virtual public C, virtual public A {
public:
	D(int i) : B(), A(1), C(i) { static A b(i); cout << "cons D" << endl; }
	bool f(int objNum) { return B::f(objNum) || C::f(objNum); }
};
int main() {
	D b(id);
	C c(), a();
	C* pc = &b;
	A* pa = pc;
	pa->f(2);
	exit(0);
}
B e(id++);
```

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

```
#include <iostream>

enum Competitiors { I, II, III, IV, V, VI };

typedef int code;

const code codes[] = { 77, 99 };

class Number
{
      int data;
public:
      Number(int arg) : data(arg) {
            (arg % 2 == 0) ? throw data : 42;
            std::cout << "init";
      }
      virtual int get() { return data; }
      ~Number() { std::cout << "tini" << data; }
}

; static int test = 1 /** <-|-> **/; static Number tset = 1;

class Player
{
      Number number;
public:
      Player(int num)
      try : number(num) {
            std::cout << "New player: " << num << std::endl;
      }
      catch (int ex) {
            std::cout << "Invalid number: " << num << std::endl;
            if (num % 4 == 0) throw codes[1];
      }

      void play(int steps) throw(code)
      try {
            if (steps > 3) throw steps;
            test += steps;
      }
      catch (int ex) {
            std::cout << "Player is stuck: " << steps << std::endl;
            if (steps % 2 == 0) throw codes[0];
      }
};

void init(int id);

int main()
{
      for (int pos = tset.get() - I; pos <= VI;)
      try {
            init(++pos);
      }
      catch (int ex) {
            std::cout << "Error code: " << ex << std::endl;
      }
      return 0;
}
```

```
void init(int id) {
      Player* player;
      switch (id) {
      case 1:
            std::cout << "Hello, Johnny" << std::endl;
      case 2:
            std::cout << "Hey there, Jane" << std::endl;
            player = new Player(id);
            if (test == tset.get())
                  case 3: std::cout << "Good luck!" << std::endl;
            test += id;
            return;
      case 4:
            do {
                  std::cout << "Testing round #" << test << std::endl;
                  Player player(id + test++);
                  if (test < 7)
                        continue;
            } while (false);
            break;
      default:
            player = new Player(test^test + id); player->play(id); return;
      case 5:
            player = new Player(id); player->play(test *= 2); break;
      case 6: try { Player best = (tset, test); best.play(6); }
                  catch (...) { init(7); }
      }
}
```

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

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

class Base {
      int x;
public:
      Base() : Base(1) { cout << "Base() " << x << endl; };
      Base(int a) : x(a) { cout << "Base(int) " << ++x << endl; };
      Base(const Base& b) : Base(b.x) { cout << "Base(Base) " << x << endl; };
      int f(int& n) { n += x >> 1;  cout << "f = " << x << endl; return x = n; }
      virtual int g(int m) { m += x << 1;  cout << "g = " << x << endl; return x = m; }
};

class Derived : public Base {
      Base b;
public:
      Derived(int a, const Base& bb) : b(bb), Base(a) { b.g(0); };
      int f(int& n) { int v = b.f(n); v += Base::f(n); return v; }
      virtual int g(int m) { int v = b.g(m); v += Base::g(m); return v; }
      Base& f(int& n, Base& bb) {
            b = b.f(n) < bb.f(n) ? b.f(n) : bb.f(n), bb; return *this; }
      int g(int m, Base bb) { return Base::g(bb.g(m)); }
};

int main()
{
      int m = 3;
      int n = 1;

      Base b;
      Derived d(1, Base(3));
      Base* pbd = &d;

      b.f(n);
      pbd->f(n);

      d.g(m);
      pbd->g(m);

      cout << d.g(1, b) << endl;
      Base &bb = d.f(n, b);

      return 0;
}
```

Task 5. 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);
      double f(double);
      int g();
      virtual int g(int i = 10);
      bool operator!=(Base& b);
      bool h(int i);
      virtual bool h(Base& b);
};

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() {
      x = f(1);
      cout << "Base::g() " << 1 << " " << x << endl;
      return x;
}

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

bool Base::operator!=(Base& b)
{
      return this->h(b);
}

bool Base::h(int i) { return x != i; }

bool Base::h(Base &b) { return x != b.x; }

class Derived : public Base {
protected:
      int x;
public:
      Derived(int a, int b) : Base(a), x(b) {};
      virtual double f(double);
      int g(int i = 20);
      bool h(Base& b);
};

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

int Derived::g(int i) {
      if (i == 10) { x = f(i << 1); }
      if (i == 20) { x = f(i << 1 | 1); }
      cout << "Derived::g(int) " << i << " " << x << endl;
      return x;
}

bool Derived::h(Base &b) { return Base::h(b) && b.h(x); }
```

```
class DerivedSunny : public Derived {
public:
      DerivedSunny(int a, int b) : Derived(--a, ++b) {};
};

class DerivedBeach : public DerivedSunny {
public:
      DerivedBeach(int a, int b) : DerivedSunny(++a, --b) {};
      int f(int);
      double f(double);
      int g(int i = 40);
      bool h(Base& b);
};

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

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

int DerivedBeach::g(int i) {
      if (i == 10) { x = f((i + x) << 1); }
      if (i == 20) { x = f((i + x) << 1 | 1); }
      if (i == 40) { x = f(((i + x) << 1 | 1) << 1); }
      cout << "DerivedBeach::g(int) " << i << " " << x << endl;
      return x;
}

bool DerivedBeach::h(Base &b) { return DerivedSunny::h(b) || b.h(Derived::x-Base::x); }

int main()
{
      Derived     d(1, 3);
      Base* pbd = new Derived(2, 4);
      DerivedSunny ds(2, 4);
      Base* pbs = new DerivedSunny(2, 4);
      DerivedBeach db(2, 4);
      DerivedSunny* psb = new DerivedBeach(2, 4);

      d.g();
      pbd->f(1);
      pbd->f(2.0);

      ds.f(2);
      ds.g();
      pbs->f(2.0);
      pbs->g(4);

      cout << (ds != d ? "Sunny" : "Cloudy") << endl;
      cout << (db != ds ? "Sunny Beach" : "Golden Sands") << endl;
      cout << (pbs != pbd ? "Sunny" : "Cloudy") << endl;
      cout << (psb != pbs ? "Sunny Beach" : "Golden Sands") << endl;

      db.g();
      psb->f(8);
      psb->g();

      return 0;
}
```

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

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

struct DaenerysTargaryen {
    string dragon = "Drogon";
    bool queen = false;
};

int throne = 1;

class LyannaStark {
public:
    LyannaStark() { throne++; }
};
class RhaegarTargaryen {
public:
    RhaegarTargaryen() { throne = throne << 1; }
};
class AegonTargaryen : public LyannaStark, virtual RhaegarTargaryen {
public:
    AegonTargaryen() { throne >> 1; }
    ~AegonTargaryen() { throne++; cout << "\nThe Lannisters send their regards"; }
};

void t(int i) {
    switch (i) {
    case 2: throw 12.7; break;
    case 9: throw LyannaStark(); break;
    case 3: throw 35.12f; break;
    case 10: throw AegonTargaryen(); break;
    case 5: throw new AegonTargaryen(); break;
    case 4: throw DaenerysTargaryen(); break;
    }
}
void g(int i) {
    switch (i) {
    case 0: throw RhaegarTargaryen(); break;
    case 6: throw 17.3f; break;
    case 11: throw 15.123; break;
    case 7: throw new AegonTargaryen(); break;
    case 8: throw AegonTargaryen(); break;
    case 1: throw 42; break;
    }
}

int main() {
    for (int o = 0; o < 12; o++) {
        cout << throne << " ";
        try {
            g(o);
            try {
                t(o);
            }
            catch (RhaegarTargaryen &rt) { cout << "|Hear Me Roar|"; }
            catch (AegonTargaryen &at) { cout << "Winter is coming"; }
            catch (double) { cout << "V@lar Dohaeri$"; }
            catch (int) { cout << "Snow"; }
            catch (DaenerysTargaryen dt) { cout << dt.dragon << " Drakaris"; }
            catch (DaenerysTargaryen *dt) { cout << "Drakaris " << dt->dragon; }
            catch (float) { cout << "Valar Morghulis/\\/\\"; }
            catch (LyannaStark *ls) { cout << "Fire and Blood"; }
            catch (AegonTargaryen *at) { cout << "/Ours is the Fury/"; }
        }
        catch (RhaegarTargaryen *rt) { cout << "|Fire And Blood|"; }
        catch (LyannaStark *ls) { cout << "*Winter is coming*"; }
        catch (float) { cout << "Valar Dohaeris"; }
        catch (double) { cout << "Valar Morghulis"; }
        catch (int) { cout << "jon"; }
        catch (AegonTargaryen at) { cout << "Here We Stand"; }
        catch (LyannaStark ls) { cout << "Hear Me Roar"; }
        catch (RhaegarTargaryen rt) { cout << "Ours is the Fury"; }
        cout << endl;
    }
    return 0;
}
```

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

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

template <class T>
class Elem {
       T val, key;
public:
       Elem<T>() { val = T(); key = T(); }
       Elem<T>(T v, T k) { val = v; key = k; }
       Elem<T> Merge(Elem<T>& el, bool bInc = false) {
              T inc = bInc ? 1 : -1;
              return Elem<T>(val + inc*el.val, key + inc*el.key);
       }
       T Add(Elem<T>& el) {
              return val * el.val + key * el.key;
       }
       T Sub(Elem<T>& el) {
              return val * el.key - key * el.val;
       }
       bool Relate(Elem<T>& el) {
              return val < el.val;
       }
       void Print() {
              cout << val << " " << key << " ";
       }
};

template <class T>
class Relation {
       Elem<T> elems[2];
public:
       Relation<T>(Elem<T>& elem1, Elem<T>& elem2) {
              elems[0] = elem1; elems[1] = elem2;
       }
       bool Product(Relation<T>& rel) {
              bool res = false;
              Elem<T> mi = elems[1].Merge(elems[0]);
              Elem<T> mj = rel.elems[1].Merge(rel.elems[0]);
              Elem<T> mk = elems[0].Merge(rel.elems[0]);
              T sij = mi.Sub(mj);
              if (sij != 0.0) {
                     T sjk = mj.Sub(mk) / sij;
                     if (sjk >= 0.0 && sjk <= 1.0) {
                            T sik = mi.Sub(mk) / sij;
                            if (sik >= 0.0 && sik <= 1.0)
                                   res = true;
                     }
              }
              return res;
       }
       bool Relate(Relation<T>& rel, int code) {
              return elems[code >> 1].Relate(rel.elems[code & 0x1]);
       }
       void Print() {
              elems[0].Print(); elems[1].Print();
       }
};

template <class T, int n>
class Connection {
       int value;
       Relation<T>* relation;
       Connection** cons;
public:
       Connection(Relation<T>* rel) {
              value = 0;
              relation = rel;
              cons = new Connection<T, n>*[n];
              for (int i = 0; i < n; i++)
                     cons[i] = NULL;
       }
```

```
	void Update(int& val) { value += val; };
	Connection<T, n>* Connect(Connection<T, n>* conpar, Relation<T>* rel, bool check, int& val)
	{
		Connection<T, n>* con = NULL;
		int ind = 1;
		if (this != NULL) {
			bool related = relation->Relate(*rel, 0x0);
			con = cons[ind++]->Connect(this, rel, check&&!related, val);
			if (!rel->Relate(*relation, 0x2)) {
				Connection<T, n>* conTmp =
					cons[ind++]->Connect(this, rel, check&&related, val);
				con = conTmp ? conTmp : con;
				if (!relation->Relate(*rel, 0x2)) {
					if (relation->Product(*rel)) {
						value++;
						val++;
					}
				}
			}
		} else if (check) {
			Connection<T, n>** selected = &conpar->cons[ind++];
			if (conpar->relation->Relate(*rel, 0x0))
				selected = &conpar->cons[ind++];
			con = *selected = new Connection<T, n>(rel);
		}
		return con;
	}
	void Print() {
		if (this != NULL) {
			relation->Print();
			cout << "| " << value << endl;
			for (int i = 0; i < n; i++)
				cons[i]->Print();
		}
	}
};

template <class T, int n>
class Collection {
	Connection<T, n>* con;
	void Insert(Elem<T>& elem1, Elem<T>& elem2) {
		Relation<T>* rel = new Relation<T>(elem1, elem2);
		if (con == NULL)
			con = new Connection<T, n>(rel);
		else {
			int value = 0;
			con->Connect(NULL, rel, true, value)->Update(value);
		}
	};
public:
	Collection() { con = NULL; };
	void Create(int arr[], int cnt) {
		for (int i = 0; i<cnt; i += 2) {
			Elem<T> elem1((arr[i] & 0xF0) >> 0x4, arr[i] & 0xF);
			Elem<T> elem2((arr[i+1] & 0xF0) >> 0x4, arr[i+1] & 0xF);
			Insert(elem1, elem2);
		}
	}
	void Print() { con->Print(); }
};

int main() {
	int nVal = 22;
	int arVal[] = { 0x77, 0xBB,  0xB7, 0xE5,  0x33, 0x74,  0x25, 0x82,  0xB4, 0xF7,
		0x32, 0x37, 0xAC, 0xD3,  0x3C, 0xEC,  0x22, 0x8C,  0xCC, 0xF6,  0x4D, 0xE3};
	Collection<double,4> collection;
	collection.Create(arVal, nVal);
	collection.Print();
	return 0;
}
```

Answers: Object oriented programming – 2019 (Sunny Beach)

Task1.

```
Competition 2018
1A
2B
3B
Competition copy from
2018
X registered for
competition!
See you next year!
X goes home :(
B goes home :(
Competition 2020
OOP 2019
A won!
4C
4C
C goes home :(
Competition 2019
OOP copy from 2019
C registered for OOP!
See you next year!
4D
D lost!
D goes home :(
See you next year!
See you next year!
```

Task2.

```
cons A (1)
cons A (4)
1 dest
cons A (-3)
cons C
cons A (-7)
4 dest
cons A (14)
cons B
cons A (221)
cons D
A::f -> 0
A::f -> 0
A::f -> 1
-3 dest
-2 dest
1 dest
```

Task3.

```
initHey there, Jane
Invalid number: 2
Error code: 2
Good luck!
Testing round #4
Invalid number: 8
Error code: 99
initNew player: 5
Player is stuck: 10
Error code: 77
Invalid number: 10
initNew player: 27
Player is stuck: 7
tini1
```

Task4.

```
Base(int) 2
Base() 2
Base(int) 4
Base(int) 2
Base(int) 5
Base(Base) 5
g = 5
f = 2
f = 2
g = 10
g = 3
g = 23
g = 9
Base(int) 3
Base(Base) 3
g = 3
g = 21
49
f = 49
f = 2
f = 27
Base(int) 42
```

Task5.

```
Derived::f(double)
Derived::g(int) 20 41
Base::f(int)
Base::f(double)
Derived::f(double)
Derived::f(double)
Derived::g(int) 20 42
Base::f(double)
Derived::g(int) 4 5
Cloudy
Sunny Beach
Sunny
Sunny Beach
DerivedBeach::f(int)
DerivedBeach::g(int) 40
186
DerivedBeach::f(double)
DerivedBeach::f(int)
DerivedBeach::g(int) 20
57
```

Task6.

```
1 Ours is the Fury
2 jon
2 V@lar Dohaeri$
2 Valar Morghulis/\/\
2 Drogon Drakaris
2 Fire and Blood
5 Valar Dohaeris
5 *Winter is coming*
11 Here We Stand
The Lannisters send their
regards
The Lannisters send their
regards
25 Hear Me Roar
26 Winter is coming
The Lannisters send their
regards
54 Valar Morghulis
```

Task7.

```
7 7 11 11 | 2
3 3 7 4 | 2
2 5 8 2 | 3
2 2 8 12 | 4
3 2 3 7 | 3
3 12 14 12 | 4
4 13 14 3 | 5
11 7 14 5 | 2
11 4 15 7 | 4
10 12 13 3 | 5
12 12 15 6 | 2
```