1 Specificați si testați funcția: **(1.5p)**

```
int f(int x) {
	if (x <= 0)
		throw std::exception("Invalid argument!");

	int rez = 0;
	while (x)
	{
		rez = rez * 10 + x % 10;
		x /= 10;
	}
	return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
int except(bool thrEx) {
	if (thrEx) {
		throw 2;
	}
	return 3;
}

int main() {
	try {
		cout << except(1 < 1);
		cout << except(true);
		cout << except(false);
	} catch (int ex) {
		cout << ex;
	}
	cout << 4;
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
  A() {cout << "A" << endl;}
  ~A() {cout << "~A" << endl;}
  void print() {
     cout << "print" << endl;
  }
};

void f() {
	A a[2];
	a[1].print();
}
int main() {
	f();
	return 0;
}
```

1 Specificați si testați funcția: **(1.5p)**

```
std::pair<int, int> f(std::vector<int> l) {
	if (l.size()<2) throw std::exception{};
	std::pair<int, int> rez{-1,-1};
	for (auto el:l) {
		if (el < rez.second) continue;
		if (rez.first < el) {
			rez.second = rez.first;
			rez.first = el;
		}else{
			rez.second=el;
		}
	}
	return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
#include <vector>
struct A {
	A() {std::cout << "A";}
	virtual void print() {
		std::cout << "A";
	}
};
struct B : public A{
	B() { std::cout << "B"; }
	void print() override {
		std::cout << "B";
	}
};
int main() {
	std::vector<A> v;
	v.push_back(A{});
	v.push_back(B{});
	for (auto& el : v) el.print();
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
	int x;
public:
	A(int x) : x{ x } {}
	 void print(){cout<< x <<endl;}
};
A f(A a) {
	a.print();
	a = A{ 10 };
	a.print();
	return a;
}
int main() {
	A a{ 4 };
	a.print();
	f(a);
	a.print();
}
```

1 Specificați si testați funcția: (1.5p)

```
bool f(int a) {
     if (a <= 0)
          throw std::exception("Illegal argument");
     int d = 2;
     while (d<a && a % d>0) d++;
     return d>=a;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Dacă sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
class A {
public:
 A(){cout << "A()" << endl;}
 void print() {cout << "printA" <<
endl;}
};
class B: public A {
public:
 B(){cout << "B()" << endl;}
 void print() {cout << "printB" <<
endl;}
};
int main() {
      A* o1 = new A();
      B* o2 = new B();
      o1->print();
      o2->print();
      delete o1;delete o2;
   return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
#include <vector>
using namespace std;
int main() {
      vector<int> v;
      v.push_back(5);
      v.push_back(7);
      v[0] = 6;
      v.push_back(8);
      auto it = v.begin();
      it = it + 1;
      while (it != v.end()) {
            cout << *it << endl;
            it++;
      }
  return 0;
}
```

1 Specificați si testați funcția: **(1.5p)**

```
bool f(int a) {
      if (a <= 1)
            throw "Illegal argument";
      int aux = 0;
      for (int i = 2; i < a; i++) {
            if (a % i == 0) {
                  aux++;
            }
      }
      return aux == 0;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <vector>
#include <iostream>
using namespace std;
class A {
public:
      virtual void f() = 0;
};
class B:public A{
public:
      void f() override {
            cout << "f din B";
      }
};
class C :public B {
public:
      void f() override {
            cout << "f din C";
      }
};
int main() {
      vector<A> v;
      B b;
      v.push_back(b);
      C c;
      v.push_back(c);
      for (auto e : v) { e.f(); }
      return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
 A() {cout << "A" << endl;}
 ~A() {cout << "~A" << endl;     }
 void print() {cout << "print" <<
endl;}
};
void f() {
      A a[2];
      a[0].print();
}
int main() {
      f();
      return 0;
}
```

1 Specificați si testați funcția: (1.5p)

```
using namespace std;
#include <vector>
#include <string>
#include <algorithm>
#include <map>
vector<int> f(vector<int> l) {
	if (l.size() == 0)
		throw exception("Illegal argument");
	map<int, int> c;
	for (auto e : l)  {
		c[e]++;
	}
	sort(l.begin(), l.end(), [&](int a, int b) {
		return c[a] > c[b]; });
	return l;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <vector>
#include <iostream>
class A {
public:
	A() {
		std::cout << "A";
	}
	virtual void print() {
		std::cout << "printA";
	}
};
class B : public A {
public:
	B() {
		std::cout << "B";
	}
	virtual void print() {
		std::cout << "printB";
	}
};
int main() {
	std::vector<A> v;
	A a;
	B b;
	v.push_back(a);
	v.push_back(b);
	for (auto e : v) {e.print();}
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
  A() {cout << "A" << endl;}
  ~A() {cout << "~A" << endl;}
  void print() {
    cout << "print" << endl;
  }
};
void f() {
	A a[2];
	a[1].print();
}
int main() {
	f();
	return 0;
}
```

3 Scrieti codul C++ ce corespunde diagramei de clase UML. **(4p)**

1 Specificați si testați funcția: **(1.5p)**

```
vector<int> f(int a) {
      if (a < 0)
            throw std::exception("Illegal argument");
      vector<int> rez;
      for (int i = 1; i <= a; i++) {
            if (a % i == 0) {
                  rez.push_back(i);
            }
      }
      return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
int except(int v) {
      if (v < 0) {throw 1;  }
      else if (v > 0){
         throw std::exception ("A");
       }
      return 0;
}
int main(){
      try {
         cout << except(1 < 1);
         cout << except(-5);
         cout << except(5);
      }catch (std::exception& e) {
            cout << "A";
      }catch (int x) {
            cout << "B";
      }
      cout << "C";
     return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
      int x;
public:
      A(int x) : x{ x } {}
       void print(){cout<< x <<",";}
};
A f(A a) {
      a.print();
      a = A{ 7 };
      a.print();
      return a;
}
int main() {
      A a{ 5 };
      a.print();
      f(a);
      a.print();
}
```

1 Specificați si testați funcția: **(1.5p)**

```
int f(int x) {
	if (x <= 0)
		throw std::exception("Invalid argument!");

	int rez = 0;
	while (x)
	{
		rez = rez * 10 + x % 10;
		x /= 10;
	}
	return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
int except(bool thrEx) {
	if (thrEx) {
		throw 2;
	}
	return 3;
}

int main() {
	try {
		cout << except(1 < 1);
		cout << except(true);
		cout << except(false);
	} catch (int ex) {
		cout << ex;
	}
	cout << 4;
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
  A() {cout << "A" << endl;}
  ~A() {cout << "~A" << endl;}
  void print() {
     cout << "print" << endl;
  }
};

void f() {
	A a[2];
	a[1].print();
}
int main() {
	f();
	return 0;
}
```

1 Specificați si testați funcția: **(1.5p)**

```
std::pair<int, int> f(std::vector<int> l) {
	if (l.size()<2) throw std::exception{};
	std::pair<int, int> rez{-1,-1};
	for (auto el:l) {
		if (el < rez.second) continue;
		if (rez.first < el) {
			rez.second = rez.first;
			rez.first = el;
		}else{
			rez.second=el;
		}
	}
	return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
#include <vector>
struct A {
	A() {std::cout << "A";}
	virtual void print() {
		std::cout << "A";
	}
};
struct B : public A{
	B() { std::cout << "B"; }
	void print() override {
		std::cout << "B";
	}
};
int main() {
	std::vector<A> v;
	v.push_back(A{});
	v.push_back(B{});
	for (auto& el : v) el.print();
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
	int x;
public:
	A(int x) : x{ x } {}
	 void print(){cout<< x <<endl;}
};
A f(A a) {
	a.print();
	a = A{ 10 };
	a.print();
	return a;
}
int main() {
	A a{ 4 };
	a.print();
	f(a);
	a.print();
}
```

1 Specificați si testați funcția: (1.5p)

```
bool f(int a) {
     if (a <= 0)
          throw std::exception("Illegal argument");
     int d = 2;
     while (d<a && a % d>0) d++;
     return d>=a;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Dacă sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
class A {
public:
 A(){cout << "A()" << endl;}
 void print() {cout << "printA" <<
endl;}
};
class B: public A {
public:
 B(){cout << "B()" << endl;}
 void print() {cout << "printB" <<
endl;}
};
int main() {
      A* o1 = new A();
      B* o2 = new B();
      o1->print();
      o2->print();
      delete o1;delete o2;
   return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
#include <vector>
using namespace std;
int main() {
      vector<int> v;
      v.push_back(5);
      v.push_back(7);
      v[0] = 6;
      v.push_back(8);
      auto it = v.begin();
      it = it + 1;
      while (it != v.end()) {
            cout << *it << endl;
            it++;
      }
  return 0;
}
```

1 Specificați si testați funcția: **(1.5p)**

```
bool f(int a) {
      if (a <= 1)
            throw "Illegal argument";
      int aux = 0;
      for (int i = 2; i < a; i++) {
            if (a % i == 0) {
                  aux++;
            }
      }
      return aux == 0;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <vector>
#include <iostream>
using namespace std;
class A {
public:
      virtual void f() = 0;
};
class B:public A{
public:
      void f() override {
            cout << "f din B";
      }
};
class C :public B {
public:
      void f() override {
            cout << "f din C";
      }
};
int main() {
      vector<A> v;
      B b;
      v.push_back(b);
      C c;
      v.push_back(c);
      for (auto e : v) { e.f(); }
      return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
 A() {cout << "A" << endl;}
 ~A() {cout << "~A" << endl;     }
 void print() {cout << "print" <<
endl;}
};
void f() {
      A a[2];
      a[0].print();
}
int main() {
      f();
      return 0;
}
```

1 Specificați si testați funcția: (1.5p)

```
using namespace std;
#include <vector>
#include <string>
#include <algorithm>
#include <map>
vector<int> f(vector<int> l) {
	if (l.size() == 0)
		throw exception("Illegal argument");
	map<int, int> c;
	for (auto e : l)  {
		c[e]++;
	}
	sort(l.begin(), l.end(), [&](int a, int b) {
		return c[a] > c[b]; });
	return l;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <vector>
#include <iostream>
class A {
public:
	A() {
		std::cout << "A";
	}
	virtual void print() {
		std::cout << "printA";
	}
};
class B : public A {
public:
	B() {
		std::cout << "B";
	}
	virtual void print() {
		std::cout << "printB";
	}
};
int main() {
	std::vector<A> v;
	A a;
	B b;
	v.push_back(a);
	v.push_back(b);
	for (auto e : v) {e.print();}
	return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
  A() {cout << "A" << endl;}
  ~A() {cout << "~A" << endl;}
  void print() {
    cout << "print" << endl;
  }
};
void f() {
	A a[2];
	a[1].print();
}
int main() {
	f();
	return 0;
}
```

3 Scrieți codul C++ ce corespunde diagramei de clase UML. **(4p)**

1 Specificați si testați funcția: **(1.5p)**

```
vector<int> f(int a) {
      if (a < 0)
            throw std::exception("Illegal argument");
      vector<int> rez;
      for (int i = 1; i <= a; i++) {
            if (a % i == 0) {
                  rez.push_back(i);
            }
      }
      return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
int except(int v) {
      if (v < 0) {throw 1;  }
      else if (v > 0){
         throw std::exception ("A");
       }
      return 0;
}
int main(){
      try {
         cout << except(1 < 1);
         cout << except(-5);
         cout << except(5);
      }catch (std::exception& e) {
            cout << "A";
      }catch (int x) {
            cout << "B";
      }
      cout << "C";
     return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
      int x;
public:
      A(int x) : x{ x } {}
       void print(){cout<< x <<",";}
};
A f(A a) {
      a.print();
      a = A{ 7 };
      a.print();
      return a;
}
int main() {
      A a{ 5 };
      a.print();
      f(a);
      a.print();
}
```

1 Specificați si testați funcția: **(1.5p)**

```
int f(int x) {
    if (x <= 0)
        throw std::exception("Invalid argument!");

    int rez = 0;
    while (x)
    {
        rez = rez * 10 + x % 10;
        x /= 10;
    }
    return rez;
}
```

2 Indicați rezultatul execuției pentru următoarele programe c++. Daca sunt erori indicați locul unde apare eroarea si motivul.

```
//2 a  (1p)
#include <iostream>
using namespace std;
int except(bool thrEx) {
    if (thrEx) {
        throw 2;
    }
    return 3;
}

int main() {
    try {
        cout << except(1 < 1);
        cout << except(true);
        cout << except(false);
    } catch (int ex) {
        cout << ex;
    }
    cout << 4;
    return 0;
}
```

```
//2 b  (0.5p)
#include <iostream>
using namespace std;
class A {
public:
  A() {cout << "A" << endl;}
  ~A() {cout << "~A" << endl;}
  void print() {
     cout << "print" << endl;
  }
};

void f() {
    A a[2];
    a[1].print();
}
int main() {
    f();
    return 0;
}
```