Enuicoc Andrei 10-03-2024
Gr. 272

9. 4

1. Specificații:

- date de intrare: număr întreg nenegativ (a)

- date de ieșire: vector cu nr. întregi cu toți divizorii lui a, exception altfel

- programul returnează divizorii unui nr. întreg nenegativ sau aruncă excepție dacă ~~p~~ parametrul nu corespunde (mai mic decât 0)

Teste:

```
~~voi~~ # include <assert>

void teste ()
{   try
    {     f(-1);
          assert(false);
    }
    catch (exception &)
    { assert(true); }
```

①

Cursice Andrei
Gr. 212

```
    auto v= f(6);
    assert(v.size()==4);
    assert(v[0]==1);
    assert(v[2]==3);
    assert(v[3]==6);
    auto x = f(2);
    assert(v.size()==2);
    auto y = f(0);
    assert(v.empty()==true);
}
```

2. a) Se va afişa: 0 B C

b) Se va afişa: 5, 5, 7, 5,

```
template <typename T>
class it;  →
```

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

4.
```
template <typename T>
class Catalog
{ private:
      string nume;
      vector <T> note;
```

Buzioc Andrei
Gr. 212

```
public:
    Catalog (const string & n): nume{n} {}
    Catalog (const string & n, const vector<T>&
      v ): nume{n}, note{v} {}
    void adaugă (const T& elem)
    { note.push_back(elem); }
```
~~Catalog operator + (const T& elem)~~
~~{ Catalog~~
```
    Catalog operator +(const T& elem)
    {  Catalog c = Catalog(this->nume,
                    this->note);

        c.adaugă(elem);

        return c; }
    Catalog & operator =(const Catalog&
                          elem)
    { if (this == &elem)
         return *this;

       this->nume = elem.nume;
       this->note = elem.note;

       return *this; }
```

Cusnir Andrei
Gr. 212

```
    friend class it<T>;
    it<T> begin(){ return it<T>(*this);}
    it<T> end(){ return it<T>(*this,
                        note.size());}
};

template <typename T>
class it
{   private:
        Catalog<T> & c;
        int poz = 0;
    public:
        it(Catalog<T> & cat) : c{cat}{}
        it(Catalog<T> & cat, int p) :
            c{cat}, poz{p} {}
        bool valid() const
        { return poz>=0 && poz<c.note.size();}
        T & element()
        { return c.note[poz];}
```

4

[illegible] Andrei
Gr. 212

```
void next()
{ poz++;}

T& operator *()
{ return element();}

it & operator ++()
{ next();
  return ~~poz~~ *this;}

~~it & operator ++()~~

it & operator +(const int& j)
{ ~~for (auto i:~~
  for(int i=0; i<j; i++)
     next();
  return ~~poz~~ *this;

bool operator ==(const it & ot)
{ return poz==ot.poz;}

bool operator != (cont it& ot)
{ return !(*this==ot);}

};
```

Cusiac Andrei
Gr. 2~~1~~2

```
3.1) class Pizza
{ private :
      int preț;



  public:

      Pizza (const int& p) : preț{p}{}

      virtual string descriere() = 0;

      int getPreț () const
       { return preț; }
   virtual ~Pizza () = default;

};

class Pizza Cu Peperoni : public Pitza
{  ~~Pizza p;~~
   Private :
       Pizza p;
```

6

Cusiac Andrei
Gr. 212

```
public:
  PizzaCuPeperoni( ~~Pizza p~~ const Pizza
        & p, const int & pret, ):
    Pizza ( pret ), p { p } {}

  string descriere() override
  { return string{"cu peperoni"}; }

  int getPret () const
  { auto i = p.getPret();
    i += 2;
    return i; }
  ~ PizzaCuPeperoni () override = default;

2)  vector < ~~Pizza~~ shared_ptr<Pizza> > p ()
  {  vector < shared_ptr < Pizza> > p;
     p.push_back ( make_shared <
```