**Exercitii Tutoriat 6**

→ Template, STL

**Cerinta: Pentru fiecare dintre programele de mai jos, spuneți dacă sunt corecte. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți ce nu este corect , de ce și ce ați corecta.**

**Partea I: Exercitii Usoare**

**Exercitiul 1**

#include<iostream>

#include<cstring>

using namespace std;

template <class T> void f(T t){cout<<1<<endl; }

template <> void f(float x){cout<<2<<endl; }

void f(float x){cout<<3<<endl; }

int main()

{

f(1);

f(2.5);

float x;

f(x);

f<>(x);

f<float>(x);

}

**Compileaza: 1 1 3 2 2**

**Exercitiul 2**

#include <iostream>

using namespace std;

template <class T>

void foo(T a, T b) {

cout << "(" << a << ", " << b << ")" << endl;

}

int main() {

const int i = 5, j = 23;

foo(2l, i);

foo(4,j);

return 0;

}

**Nu compileaza:**

**no instance of function template "foo" matches the argument list**

**no matching function for call to 'foo(long int, const int&)'**

**2l → converteste 2 int la 2 long.**

**Template cere acelasi tip de date T (int != long)**

**Solutie:** foo(2, i);

**Exercitiul 3**

#include<iostream>

using namespace std;

template<class X>

int functie(X \*x, int y) { return x+y; }

int functie(int \*x, int y) { return x-y; }

int main() {

int \*a= new int(20), b=63;

cout<<functie(a,b);

return 0;

}

**Nu compileaza: Functia functie() are tipul int, dar intoarce int\* care nu se poate converti la int**

**Solutie:** int functie(int \*x, int y) { return \*x-y;}

**Partea II: Exercitii Medii**

**Exercitiul 1**

#include <iostream>

using namespace std;

template <typename T> class Test

{

T x;

public:

Test(T a) : x(a) {}

template <class U> friend ostream& operator<<(ostream& out, const Test<U>& b);

};

template <class U> ostream& operator<<(ostream& out, const Test<U>& b)

{

out << b.x;

return out;

}

int main() {

Test<int> b(20);

cout << b << endl;

return 0;

}

**Compileaza : 20**

**Exercitiul 2**

#include <iostream>

using namespace std;

class Test

{

const int x;

public:

Test(int a = 0) : x(a) {

cout << "Test(" << x << ")" << endl;

}

friend ostream& operator<<(ostream& out, const Test& b) {

out << b.x;

return out;

}

};

template <class T> void functie(T ob) {

T copy;

cout << copy=ob << endl;

}

int main() {

Test ob(13);

functie(ob);

return 0;

}

**Nu compileaza : no match for 'operator<<' (operand types are 'Test' and '<unresolved overloaded function type>')**

**Modificare :**  cout << copy << endl;

**Cateva explicatii: Nu putem face atribuirea copy = ob (operator egal) pentru ca data membra x este const. Chiar daca x nu ar fi fost const am fi avut eroare din cauza :** cout << copy = ob << endl; **Aici compilatorul intelege : (**cout << copy = ob) << endl; **- adica incearca sa atribuie unui &ostream un Test. Corect ar fi fost** cout << (copy = ob) << endl;

**Exercitiul 3**

#include<iostream>

#include<cstring>

using namespace std;

template <class T> T maxim (T a, T b)

{

cout<<"template"<<endl;

if(a>b) // ceva dubios aici?

return a;

return b;

}

template <> const char \* maxim(const char\* a,const char\* b)

{

cout<<"supraincarcare const"<<endl;

if (strcmp(a,b)>0)

return a;

return b;

}

template <> char \* maxim ( char\* a, char\* b)

{

cout<<"supraincarcare neconst"<<endl;

if (strcmp(a,b)>0)

return a;

return b;

}

int main()

{

char v1[10]="abc", v2[10]="bcd";

cout<<maxim("ab","bc")<<endl;

cout<<maxim(v1,v2)<<endl;

cout<<maxim<char\*>(v1,"ab");

return 0;

}

**Compileaza:**

**supraincarcare const**

**bc**

**supraincarcare neconst**

**bcd**

**supraincarcare neconst**

**abc**

**Partea III: Exercitii “Grele”**

**Exercitiul 1**

#include <iostream>

using namespace std;

template <typename T> class A;

template <typename T> ostream& operator<<(ostream& out, const A<T>& a);

template <typename T> class A

{

T x;

public:

A(T i = 0) : x(i) {}

A operator+(const A& a) const { return A(x + a.x); }

friend ostream& operator<< (ostream& out, const A<T>& a);

};

template <typename T> ostream& operator<<(ostream& out, const A<T>& a) {

return out << a.x;

}

int main() {

A<int> a1(5), a2(10);

cout << a1 + a2 << endl;

return 0;

}

**Nu compileaza:**

**friend declaration 'std::ostream& operator<<(std::ostream&, const A<T>&)' declares a non-template function [-Wnon-template-friend]**

**corect:**

friend ostream& operator<< <> (ostream& out, const A<T>& a);

**Exercitiul 2 -mai ales pentru cine a fost la curs ;)**

#include <iostream>

class Test {

int a;

public:

Test(int x = 0) : a(x) {}

};

template <class T>

void sorteaza(T v[], int n) {

for (int i = 0; i < n - 1; i++)

for (int j = i + 1; j < n; j++)

if (v[i] > v[j]) std::swap(v[i], v[j]);

}

int main() {

Test v[] = {3, 1, 2};

sorteaza(v, 3);

}

**Nu compileaza: no match for 'operator>' (operand types are 'Test' and 'Test')**

**Nu s-a redefinit operatorul “>” - compilatorul nu stie dupa ce anume sa faca compararea**

**Exercitiul 3**

#include<iostream>

#include<vector>

class SimpleClass

{

int m\_aMember;

public: SimpleClass(int i = 10)

{ std::cout << "C"<< std::endl;

m\_aMember = i; }

SimpleClass(const SimpleClass &t)

{ std::cout << "CC" << std::endl; }

~SimpleClass() {

std::cout << "D" << std::endl; }};

int main(){

std::vector<SimpleClass> vec;

for(int i = 0; i < 2; i++) vec.push\_back(i);

return 0;

}

**Compileaza:**

C

CC

D

C

CC

CC

D

D

D

D