**Exercitiul 6**

**var 1**

#include <iostream>

using namespace std;

class A{

int x;

public:

A(int i = 25){x = i; }

int& get\_x() const { return x; }

void set\_x(int i) { x = i; }

A& operator=(A a1){

set\_x(a1.get\_x());

return \*this;

}

};

int main()

{

A a(18), b(7);

(b=a).set\_x(27);

int i;

i=b.get\_x();

return 0;

}

//Nu compileaza 0.1p

//Nu putem returna o referinta dintr-o functie constanta

//(s-ar putea modifica ulterior valoarea respectiva dupa get\_x()) 0.2p

//Pentru a compila fie scoatem identificatorul const

//fie nu mai returnam int& ci returnam doar int (scoatem &) 0.2p

------------------------------------------------------

**var 2**

#include <iostream>

using namespace std;

class A{

int x;

public:

A(int i = 25){ x = i; }

int get\_x() const{ return x;}

void set\_x(int i){ x=i; }

A operator=(A a1)

{

set\_x(a1.get\_x());

return a1;

}

};

int main()

{

A a(18), b(7);

(b=a).set\_x(27);

int i;

i=b.get\_x();

return 0;

}

//Compileaza si da valoarea 18 lui i

// 0.1 pentru ca copileaza si ceva explicatii

// 0.2 p pentru 27 si ajunge la 0.3 p

// 0.4 p pentru 18 si ajunge la 0.5 p

// alte valori (7, 25) 0.1 si ajunge la 0.2 p

-------------------------------------------------------

**var 3**

#include <iostream>

using namespace std;

class A

{

int \*x;

public:

A(int i = 25){ x = new int(i); }

int& get\_x() const { return \*x; }

void set\_x(int i) { x = new int(i); }

A& operator=(A a1)

{

set\_x(a1.get\_x());

return \*this;

}

};

int main()

{

A a(18), b(7);

(b=a).set\_x(27);

int i;

i=b.get\_x();

return 0;

}

//Compileaza si se afiseaza 27

//0.1p ca compileaza si alte explicatii

// 0.2p pentru explicatia de ce b va avea valoarea 27 si nu 18 functia = intaorce referinta etc

// 0.2p pentru 18

// daca zice ca afiseaza 18 are 0.3p

// daca zice alte valori apropiate: 25, 7 0.2p

-------------------------------------------------------

**var 4**

#include <iostream>

using namespace std;

class A {

int \*x;

public:

A() { x = new int(0); }

A(int i) { x = new int(i); }

int& get\_x() const { return \*x; }

void set\_x(int i) { x = new int(i); }

A operator=(A a1) { set\_x(a1.get\_x());return a1;}

};

class B : public A {

int y;

public:

B() : A() { y = 0; }

B(int i) : A(i){ y = i;}

void afisare() const { cout << y; }

};

int main()

{

B a(112), b, \*c;

int i;

i=(b = a).get\_x();

(c = &a)->afisare();

return 0;

}

//Compileaza si afiseaza

//112 si i va avea valoarea 112

// 0.1 pentru ca compileaza si descrierea variabilelor

// 0.2p pentru justificarea ca i ia valoarea 112

// 0.2 p pentru afisarea pe ecran a lui 112

-------------------------------------------------------

**var 5**

#include <iostream>

using namespace std;

class A {

int x;

public:

A(){ x = 0; }

A(int i) { x = i; }

int& get\_x() const { return x; }

void set\_x(int i) { x = i; }

A operator=(A a1) { set\_x(a1.get\_x()); return a1;}

};

class B : public A {

int y;

public:

B() : A(){ y = 0;}

B(int i) : A(i) { y = i; }

void afisare() const { cout << y; }

};

int main()

{

B a(112), b, \*c;

int i;

i= (b = a).get\_x()<<"\n";

(c = &a)->afisare();

return 0;

}

//Nu compileaza 0.1p

//Nu putem returna o referinta dintr-o functie constanta

//(s-ar putea modifica ulterior valoarea respectiva dupa get\_x()) 0.2p

//Pentru a compila fie scoatem identificatorul const

//fie nu mai returnam int& ci returnam doar int (scoatem &) 0.2p