Problema 9:

Varianta 1: NU COMPILEAZA linia 11 da eroare: pointerul p este constant deci nu poate apela o functie neconst

#include <iostream>

using namespace std;

class X

{ int i;

public: X(int j=10)

{ i=j; cout<< i<< " ";}

const int afisare(int j)

{ cout<<i<< " "; return i+j; }; };

int main()

{ const X O (7), &O2=O, \*p=&O2;

cout<<p->afisare(6);

return 0; }

ca spune ca nu compileaza cu o justificare rezonabila 0.1p

ca linia 22: cout<<p->afisare(6); da eroare pointerul p este constant deci nu poate apela o functie neconst 0.2p

daca nu justifica bine modificarea nu conteaza (nu se puncteaza urmatoarea parte)

modificarea care il face sa mearga: afisare se defineste const la linia 13: 0.2p

Varianta 2: NU COMPILEAZA linia 11 da eroare: pointerul p este constant deci nu poate apela o functie neconst

#include <iostream>

using namespace std;

class X

{ int i;

public: X(int j=10)

{ i=j; cout<< i<< " ";}

int const afisare(int j)

{ cout<<i<< " "; return i+j; }; };

int main()

{ const X O (7), &O2=O, \*p=&O2;

cout<<p->afisare(6);

return 0; }

ca spune ca nu compileaza cu o justificare rezonabila 0.1p

ca linia 22: cout<<p->afisare(6); da eroare pointerul p este constant deci nu poate apela o functie neconst 0.2p

daca nu justifica bine modificarea nu conteaza (nu se puncteaza urmatoarea parte)

modificarea care il face sa mearga: afisare se defineste const la linia 13: 0.2p

Varianta 3: COMPILEAZA si afiseaza si afiseaza valorile 1 1 7

#include <iostream>

using namespace std;

class X

{ int i;

public: X(int j=10)

{ i=j; cout<< i<< " ";}

int afisare(int j) const

{ cout<<i<< " "; return i+j; }; };

int main()

{ const X O (1), &O2=O, \*p=&O2;

cout<<p->afisare(6);

return 0; }

spune ca compileaza 0.1p (daca spune doar ca compileaza nu primeste nimic)

ca afiseaza 117 0.2p (daca spune ca compileaza si 1 1 7 fara altceva 0.1p)

Se apeleaza mai intai constructorul X cu parametrul 1 si se afiseaza 1

Apelam functia afisare prin pointer-ul p si vom afisa i-ul corespunzator pentru valoarea la care arata p, adica 1, iar apoi se intoarce spre cout adunarea 1 + 6 care are ca rezultat afisarea lui 7 pe ecran 0.2p

Varianta 4: COMPILEAZA si afiseaza si afiseaza valorile 7 7 13

#include <iostream>

using namespace std;

class X

{ int i;

public: X(int j=10)

{ i=j; cout<< i<< " ";}

int afisare(int j) const

{ cout<<i<< " "; return i+j; }; };

int main()

{ const X O (7), &O2=O, \*p=&O2;

cout<<p->afisare(6);

return 0; }

spune ca compileaza 0.1p (daca spune doar ca compileaza nu primeste nimic)

ca afiseaza 7 7 13 0.2p (daca spune ca compileaza si 7 7 13 fara altceva 0.1p)

Se apeleaza mai intai constructorul X cu parametrul 7 si se afiseaza 7

Apelam functia afisare prin pointer-ul p si vom afisa i-ul corespunzator pentru valoarea la care arata p, adica 7, iar apoi se intoarce spre cout adunarea 7 + 6 care are ca rezultat afisarea lui 13 pe ecran 0.2p

Varianta 5: COMPILEAZA si afiseaza valoarea 1 1 -5

#include <iostream>

using namespace std;

class X

{ int i;

public: X(int j=10)

{ i=j; cout<< i<< " ";}

int afisare(int j) const

{ cout<<i<< " "; return i-j; }; };

int main()

{ const X O (1), &O2=O, \*p=&O2;

cout<<p->afisare(6);

return 0; }

spune ca compileaza 0.1p (daca spune doar ca compileaza nu primeste nimic)

ca afiseaza 11-5 0.2p (daca spune ca da si 1 1 -5 fara altceva 0.1p)

Se apeleaza mai intai constructorul X cu parametrul 1 si se afiseaza 1

Apelam functia afisare prin pointer-ul p si vom afisa i-ul corespunzator pentru valoarea la care arata p, adica 1, iar apoi se intoarce spre cout scaderea 1 - 6 care are ca rezultat afisarea lui -5 pe ecran 0.2p