1. -12
2. Nu compileaza datorita declararii B b unde este nevoie de un constructor in B fara parametrii sau cu parametrii cu valori implicite. Modific B(int i=0): A(i).Afiseaza 112112
3. Nu compileaza deoarece la transmiterea obiectului ca al doilea parametru al fucntiei ostream & operaro<< se apeleaza copy constructor-ul care nu poate primi un obiect const, deoarece este parametru transmis cu referinta si orice modificare asupra lui se vede si asupra lui const B b, care incalca proprietatile unui obiect costant . Deci ca sa compileze modific B(const B & ob).
4. 0 0
5. cb cd1 cd2 cb cd1 cb cd1 6 dd1 db dd1 db dd2 dd1 db dd1 db dd2 dd1 db
6. 1
7. depinde de compilator (x+y are ca rezultat o adresa in x) este o adresa la care adun intregul y=23 deci ma mut in memorie la adresa x +23 aici afalandu-se 0(valoare aleatorie)
8. Nu compileaza deoarece upcastingul este posibil doar cu mostenire public intre clase.

Schimb tipul de mostenire al clasei D in public B si va afisa 1.

1. Nu compileaza deoarece nu este posibila conversia D \*p=new B. C programul sa functioneze transform in B \*p=new D care este un upcatsing correct si va afisat 8027.
2. : Nu compileaza deoarece o functie constanta nu are voie sa returneze o adresa deoarece orice modificare asupra rezultatului lui f se va vedea asupra pointerului this.Ca pprogramul sa compileze modific in int f() const ;
3. nu compileaza deoarece instructiunea x=f(x) va da eroare deoarece nu stie pe care dintre functiile f sa le apeleze.Solutia ar fi sa modific tipul parametrului din a 3 a definitie a functiei f in float y.Rezultatul depinde de valoarea introdusa.
4. EROARE la rulare. Nu este corecta definitia metodei f, deoarece predispune la modificare un obiect care este creat cu alt scop fata de cel al lui main, practice la iesirea din f se distruge. Programul compileaza dar nu afiseaza nimic. Corectez prin eliminarea & din definitia metodei f : cls1 f().
5. -11
6. Nu compileaza deoarece nu se poate realiza conversia D \*p2=new B. Ca programul sa compileze modific declaratia pointerului p2 in D \*p2=new D, afiseaza 41.
7. Nu compileaza deoarece in B f(B b) incearca sa se acceseze prin ob.x un camp protected din ob.Modific in public accesul tuturor membrilor din clasa B.Afiseaza -5.
8. 2030
9. Nu compileaza deoarece la upcasting se apeleaza functiile nervituale din clasa B si cum f1 nu e functie in B nu se poate apela prin intermediul pointerului p.
10. nu compileaza deoarece se incearca accesarea functiei f1 din clasa B unde nu exista

O posibila rezolvare este transformarea pointerilor astfel incat sa se produca in apel corect al functiei f1 din derivata D: cout << ((D\*)p)->f1(\*((B\*)p)).Afiseaza 1170.

1. Nu compileaza deoarece nu exista un constructor fara parametrii sau unul cu parametru implicit pe care sa-l apeleze atunci cand declara vectorul p. Eroarea provine de la linia cls \*p=new cls[10]. Modific parametrul constructorului in int i=0 . Afiseaza 0123456789
2. 0
3. nu compileaza in primul rand ca f este o metoda a clasei cls si trebui apelata printr-o instant a ei in al doilea rand d este un obiect constant deci poate lucre numai cu metode constante si in al treilea rand cls & c nu poate primi la apel un obiect constant .Ca programul sa compileze scot const din declararea obiectului d si apelez d.f(d).Afiseaza 3.
4. Nu compileaza deoarece mister2 primeste parametru cu referinta iar obiectul returnat de mister1 este constant deci se incalca definitia const-ului . Ca programul sa functioneze elimi & din definitia lui mister2.Afiseaza starea curenta 6 endl starea curenta 7.
5. Nu compileaza deoarece un obiect constant nu poate apela o metoda neconstanta.Ca programul sa compileze adauga cuvantul cost intre antetul metodei operator[] si {.
6. Nu compileaza deoarece un obiect constant nu poate apela o metoda neconstanta. Ca programul sa compileze elimin const din instantierea obiectului : cls d(15).Afiseaza 15
7. Nu compileaza deoarece o variabila statica nu poate fi modificata in lista de initializare a constructtorilor. Ca programul sa compileze scot y(j) si pun in definitia constructorului lui A y=j; .Afiseaza 2.
8. 3
9. 7
10. -16
11. 200
12. programul nu compileaza din cauza instructiunii D::get\_x() deoarece metodat get\_x() nu poate fi apelata vara o instanta a clasei D. Ca programul sa compileze modific apelul prin pointerul p->get\_x() si afiseaza 11bbbbbbbbbb3b1.
13. programul compileaza si va afisa o adresa deoarece intra pe prima definitie a functie fun si va aduna un pointer cu un intreg=> pointer
14. x=20, y=22
15. Nu compileaza din cauza ca incearca in functia f din clasa B sa acceseze x-ul lui a prin a.x care e protected.Ca programul sa compileze modificam specificatorul de acces al lui x in public in clasa A.Afiseaza -12.
16. Programul nu compileaza deoarece se incearca accesarea unui membru static care nu a fost declarant.Ca programul sa compileze adaugam declaratia lui x dupa clasa A : int A:: x. Afiseaza 0(deoarece x este static si apartine de tip nu de instanta).
17. Nimic
18. 9 8 7 6 5 4 3 2 1 0 (pe cate un rand nou)
19. nu compileaza deoarece exista cele 2 definitii ale lui test : test(int e, int f) si test(int &c, int &d) pe care nu va stii sa intre apelul test(a,b) dand o eroare de ambiguitate.Ca programul sa compileze apelam explicit functia template test<int>(a,b).Afiseaza ttest si test3 pe 2 linii diferite.
20. Programul nu compileaza deoarece se incearca accesare unui membru static care nu a fost initializat. Ca sa compileze adaug dupa definitia clasei A : int A:: x. Totusi va da eroare la rulare deoarce nu este corecta definitia metodei set\_x ca acesta sa returneze o adresa, ar fi trebuit ca y sa fie de tip pointer, cee ace nu e . Scot & din definitia metodei si va afisa valoarea 7.
21. -3
22. Nu compileaza deoarece obiectul o este accesibil din exteriorul instructiunii if else deci eroarea va aparea la linia cout<<o.get\_i(). Ca programul sa compileze pot sa sterg instructiunea de afisare sau sa o adaug atat in partea de la if cat si in blocul de instructiuni de la else.
23. : nu compileaza deoarece functia f() fara niciun parametru nu este definita.Ca programul sa compileze inlocuiesc return f()/2 din definitia functiei g cu return 2.Afiseaza valoarea 2.
24. 22345
25. Nu compileaza deoarece operatorul +=nu e supraincarcat in clasa B. Ca programul sa compileze , avand in vedere ca in clasa B este supraincarcat operatorul + modific a+=b cu a=a+b.Afiseaza 21.
26. 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
27. Nu compileaza deoarece metoda din clasa D get\_x() incearca accesarea lui care este privat in clasa de baza deci inaccesibil in clasa derivata.Ca programul sa compileze modific accesul lui x din private implicit in protected si va fi afisata o valoarea aleatoare a lui x deoarece se face atribuirea v=x nu x=v .
28. 7 7 7 7 14
29. 7 7 7 (pe cate o linie noua)
30. 1131
31. 6 21 3
32. 6 (Nu se apeleaza supraincarcarea operatorului + ci operatia b+=a este permisa datorita supraincarcarii operatorului int (), astfel ca a+=b nu este posibila.)
33. compileaza, dar da eroare la rulare datorita definitiei functiei f. Incearca sa extraga ce se afla la adresa lui c.vi, ceea ce ar fi fost posibil daca c.vi era pointer.Daca elimin & din definitia functiei si scot aplicarea operatorului = pe f(d) care e un obiect temporar atunci programul va afisa valoarea 15.
34. 41
35. nu compileaza deoarece i nu este accesibil in constructorul clasei D deoarece este private implicit in clasa de baza.Ca programul sa compileze modific acces in protected: int i.Afiseaza valoarea 2.
36. nu compileaza deoarece x nu e prietena cu z chiar daca x este prietena cu y si y este cu z, prietenia intre clase nu este tranzitiva, prin urmare in clasa z functia f nu are acces la x care este private .Ca programul sa compileze modific accesul lui x din private implicit in public in clasa x. Afiseaza 0.
37. 1
38. -15
39. 1500
40. 11315
41. -17
42. nu compileaza datorita mosteniri private implicita a lui cls2 din cls1 astfel functia get\_x() nu mai este accesibila printr-o instant a lui cls2 din afara clasei.Ca programul sa compileze modific tipul de mostenire cls2: public cls1 si afiseaza valoarea 37
43. 4
44. Nu compileaza datorita mostenirii de tip private implicit a lui B din A astfel ca C nu mai are acces la metoda get\_x() .Ca programul sa compileze modificam tipul de mostenire a lui b in class B: public A.Afiseaza valoarea 25.
45. Nu compileaza deoarece functia f returneaza ce se afla la adresa lui x in intregul b.Ca programul sa functioneze elimin & din definitia lui f.Afiseaza -3.
46. afiseaza pe prima pozitie o valoare aleatoare urmata de 000-150000.
47. 20 22
48. Nu compileaza deoarece functia operator+ din clasa D nu are acces la x pentru ca este private implicit.Ca programul sa compileze modific accesul lui x din clasa B in protected.Afiseaza -5.
49. 28
50. nu compileaza deoarece datorita mostenirii implicite private nu mai avem acces la metoda get\_x printr-o instant a cls2.Ca programul sa compileze modificam tipul de mostenire in class cls2: public cls1.Afiseaza valoarea 37.
51. Nu compileaza deoarece nu se pot aduna doi pointeri .Ca programul sa compileze modific in definitia functiei template return x+(\*y) deoarece \*y va returna un intreg iar adunarea dintre un intreg si un pointer este posibila avand ca rezultat tot un pointer (aici int \*).afiseaza valoearea 0 (garbage value).
52. 5
53. tipuri diferite .(P1A!=P1B)
54. Nu compileaza deoarece ob.f2() este un obiect temporar care trebuie sa apeleze o functie constanta .Ca programul sa compileze adaug cuvantul const la definitia lui f1 : int f1()const {..}.Afiseaza 23.