Să se implementeze clasa ***Student*** (***codStud*** – cod reprezentat de un număr întreg constant, ***nrNote*** – număr note, ***note*** – vector cu dimensiune variabilă, alocat dinamic, cu numere întregi ce reprezintă notele studentului, ***nrPctCredit*** – variabilă întreagă statică ce reprezintă numărul de puncte credit pe care le obține un student pentru fiecare nota >=5, inițializat cu 3) și să se dezvolte aplicația C++ care conține următoarele instrucțiuni din main():

|  |  |
| --- | --- |
|  | Definire clasa si atribute conform cerințelor |
|  | Implementare destructor pentru a evita generarea de memory leaks |
|  | **void main()**  **{**  Student s1(102);  cout << s1.getNrNote(); //afișează 0  int note[]{ 10,7,8,10,4 };  Student s2(104, note, 5);  cout << "\n" << s2.getNrNote(); //afișează 5 |
|  | Student s3 = s2;  cout << "\n" << s3.getCod(); //afișează 104 |
|  | int note2[]{ 5,5,4,10 };  s1.setNote(note2,4);  cout <<"\n"<< s1.getNrNote(); //afișează 4 |
|  | Student::nrPctCredit = 4;  s3 = s1;  bool promovat = s3.estePromovat(15); //15 reprezintă nr minim de puncte credit pentru a fi promovat  if (promovat)  cout << "\nStudentul s3 are un numar suficient de puncte credit";  else  cout << "\nStudentul s3 nu are un numar suficient de puncte credit"; |
|  | Student\* ps1 = new Student(109);  \*ps1 = s1;  cout << \*ps1; //afișează Studentul cu codul 109 are 4 note: 5 5 4 10 |
|  | **... //** dezalocă spațiul ocupat de ps1 **–** implementați apelul pentru ștergere  float medie = (float)s1;  cout << "\n"<<medie; //afișează 6 |
|  | s1 = s3++; //dublează ultima nota  cout << s1; //afișează Studentul cu codul 102 are 4 note: 5 5 4 10  cout << s3; //afișează Studentul cu codul 104 are 5 note: 5 5 4 10 10 |
|  | if (s1 == s3)  cout << "\ns4 are acelasi numar de note de trecere cu s3";  else  cout << "\ns4 nu are acelasi numar de note de trecere cu s3";  //se va afișa mesajul corect |