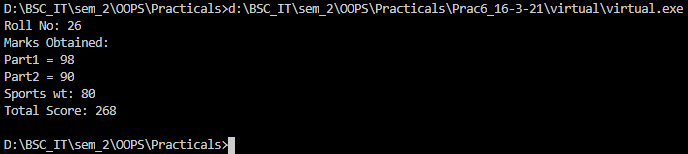
Practical 6: Virtual Class

1. Implement Virtual Base Class

Code:

|  |
| --- |
| // Implement Virtual Base Class  #include <iostream>  using std::cout;  using std::endl;  //------------(Grand Parent) Student class----------------------  class student  {  protected:      int roll\_number;  public:      void get\_number(int a)      {          roll\_number = a;      }      void put\_number(void)      {          cout << "Roll No: " << roll\_number << endl;      }  };  // ----------(Parent) Test Class------------------  class test : virtual public student  {  protected:      float part1, part2;  public:      void get\_marks(float x, float y)      {          part1 = x;          part2 = y;      }      void put\_marks(void)      {          cout << "Marks Obtained: " << endl               << "Part1 = " << part1 << endl               << "Part2 = " << part2 << endl;      }  };  // ----------(Parent) Sports Class------------------  class sports : public virtual student  {  protected:      float score;  public:      void get\_score(float s)      {          score = s;      }      void put\_score(void)      {          cout << "Sports wt: " << score << endl;      }  };  // ----------(Child) Result Class------------------  class result : public test, public sports  {  private:      float total;  public:      void display()      {          total = part1 + part2 + score;          put\_number();          put\_marks();          put\_score();          cout << "Total Score: " << total << endl;      }  };  //-----Main Program-----------  int main()  {      result student\_1;      student\_1.get\_number(26);      student\_1.get\_marks(98, 90.0);      student\_1.get\_score(80.0);      student\_1.display();      return 0;  } |

Output:

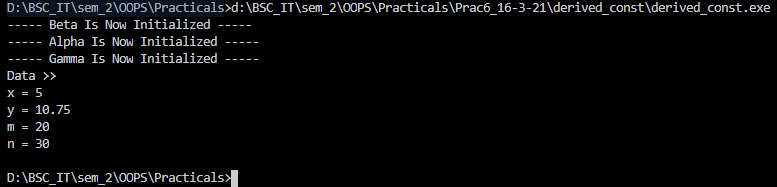


1. Implement Constructors in Derived Class

Code:

|  |
| --- |
| //B. Implement Constructors in Derived Class  #include <iostream>  using std::endl;  using std::cout;  //------------------Alpha class-----------------------  class alpha  {  private:      int x;  public:      alpha(int i)      {          x = i;          cout << "----- Alpha Is Now Initialized -----" << endl;      }      void show\_x()      {          cout << "x = " << x << endl;      }  };  //------------------Beta class-----------------------  class beta  {  private:      float y;  public:      beta(float j)      {          y = j;          cout << "----- Beta Is Now Initialized -----" << endl;      }      void show\_y()      {          cout << "y = " << y << endl;      }  };  //------------------Gamma class-----------------------  class gamma : public beta, public alpha  {  private:      int m, n;  public:      gamma(int a, float b, int c, int d) : beta(b), alpha(a)      {          m = c;          n = d;          cout << "----- Gamma Is Now Initialized -----"<< endl;      }      void show\_mn()      {          cout << "m = " << m << endl;          cout << "n = " << n << endl;      }  };  //------------------Main Program-----------------------  int main()  {      gamma g(5, 10.75, 20, 30);      cout << "Data >> " << endl;      g.show\_x();      g.show\_y();      g.show\_mn();      return 0;  } |

Output:



Write-up:

