

**LAB 5**

Submitted by:

Name: Haseeb Ullah

ID No: F20232661009

Section: V12

OOPs

Submitted to:

M. OWAIS KHAN

**SHALLOW AND DEEP CONSTRUCTOR**

Date : 03/25/2023

C-II Block C 2 Phase 1 Johar Town, Lahore, Punjab 54770.

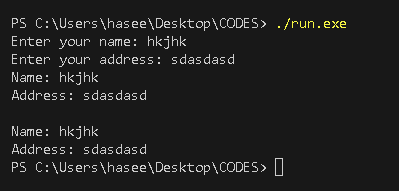
**Problem Statement 1:**

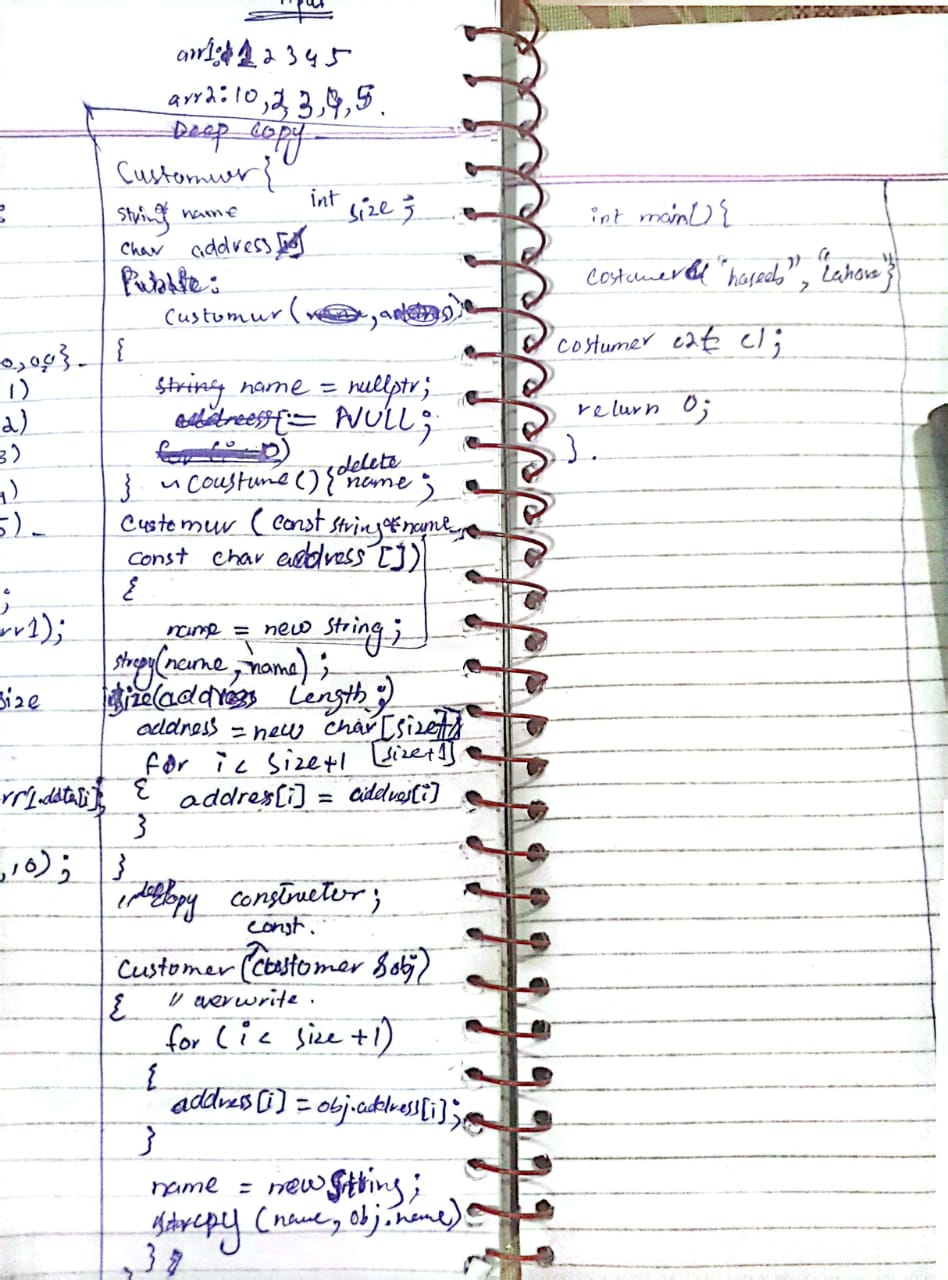
Implement a copy constructor for a class that represents a customer. The Customer class has two member variables: "name", which is a dynamically allocated string, and "address", which is a statically allocated char array. Test your implementation by creating two instances of the Customer class and using the copy constructor to create a deep copy of one of them.

**SOURCE CODE**

|  |
| --- |
| // Problem Statement 1:  // Implement a copy constructor for a class that represents a customer.  // The Customer class has two member variables: "name", which is a dynamically allocated string, and "address",  // which is a statically allocated char array.  // Test your implementation by creating two instances of the Customer class and  // using the copy constructor to create a deep copy of one of them.  #include <iostream>  #include <string.h>  using namespace std;  class Customer  {  string \*name;  char address[20];  public:  Customer(string cname, char caddress[])  {  name = new string(cname);  // name = cname;  strcpy(address,caddress);    }  ~Customer()  {  delete name;  }  Customer(const Customer &obj )  {  name = new string;  name = new string(\*obj.name);  // name = obj.name;  strcpy(address, obj.address);  }  void display()  {  cout << "Name: "<<\*name << ", Address: " << address <<endl;  }  };  int main()  {  char address[] = "Lahore";  Customer c1("Haseeb", address);  c1.display();  Customer c2 = c1;  c2.display();  return 0;  } |

**OUTPUT**

****

****

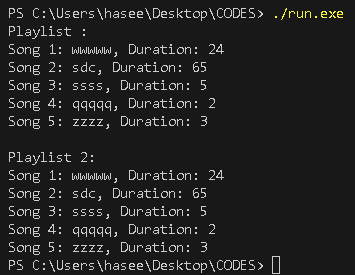
**Problem Statement 2:**

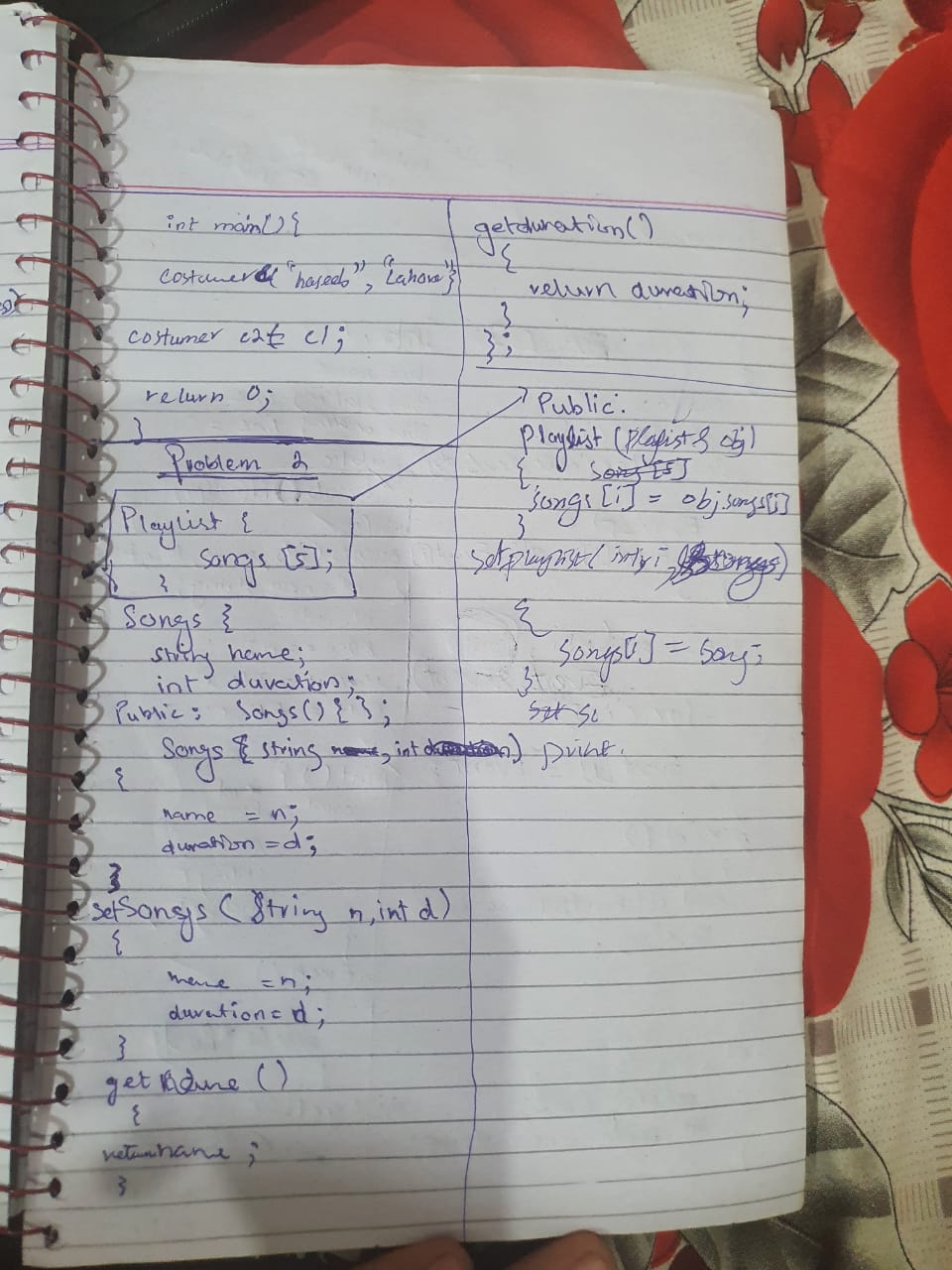
Implement a shallow copy constructor for a class that represents a playlist. The Playlist class contains an array of songs which has a name and a duration. Test your implementation by creating two instances of the Playlist class and using the copy constructor to create a shallow copy of one of them.

**SOURCE CODE**

|  |
| --- |
| // Problem Statement 2:  // Implement a shallow copy constructor for a class that represents a playlist.  // The Playlist class contains an array of songs which has a name and a duration.  // Test your implementation by creating two instances of the Playlist class and  // using the copy constructor to create a shallow copy of one of them.  #include <iostream>  using namespace std;  class Songs  {      string name;      int duration;  public:      Songs(){}      Songs(string n, int d)      {          name = n;          duration = d;      }      void setsongs(string n, int d)      {          name = n;          duration = d;      }      string get\_name()      {          return name;      }      int get\_duration()      {          return duration;      }  };  class Playlist  {      Songs songs[5];  public:      Playlist() {}      Playlist(Playlist& obj)      {          for (int i = 0; i < 5; i++)          {              songs[i] = obj.songs[i];          }        }      void set\_song(int i, const Songs& song)      {          songs[i] = song;      }      void print()      {          for (int i = 0; i < 5; i++)          {              cout << "Song " << i + 1 << ": " << songs[i].get\_name() << ", Duration: " << songs[i].get\_duration() << endl;          }      }  };  int main()  {      Playlist playlist1;      playlist1.set\_song(4 ,Songs("zzzz",3));      playlist1.set\_song(3 ,Songs("qqqqq",2));      playlist1.set\_song(0 ,Songs("wwwww",24));      playlist1.set\_song(2 ,Songs("ssss",5));      playlist1.set\_song(1 ,Songs("sdc",65));      Playlist playlist2 = playlist1;      cout << "Playlist :" << endl;      playlist1.print();      cout << "\nPlaylist 2:" << endl;      playlist2.print();        return 0;  } |

**OUTPUT**

****



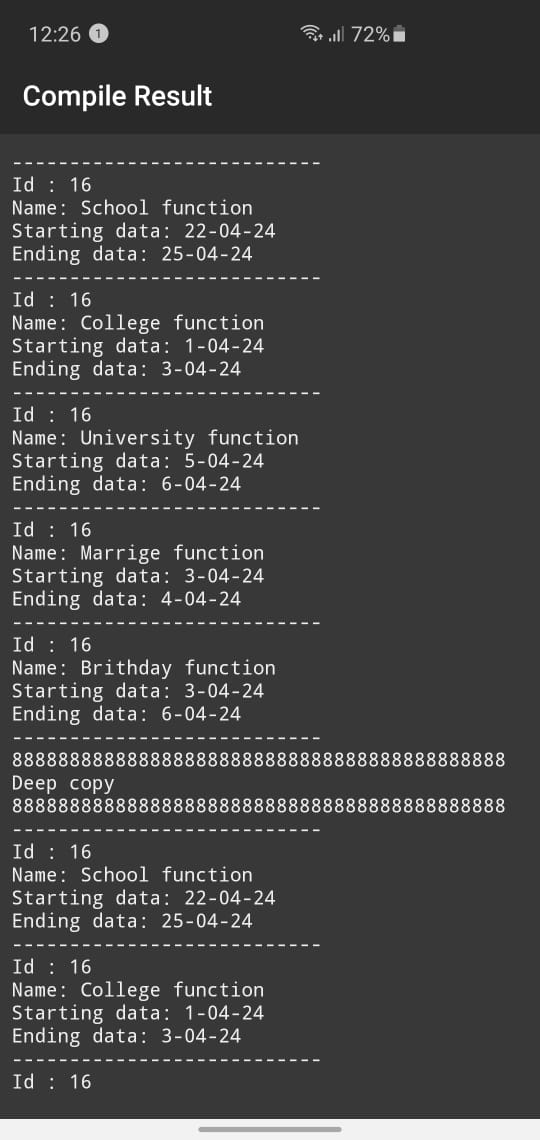
**Problem Statement 3:**

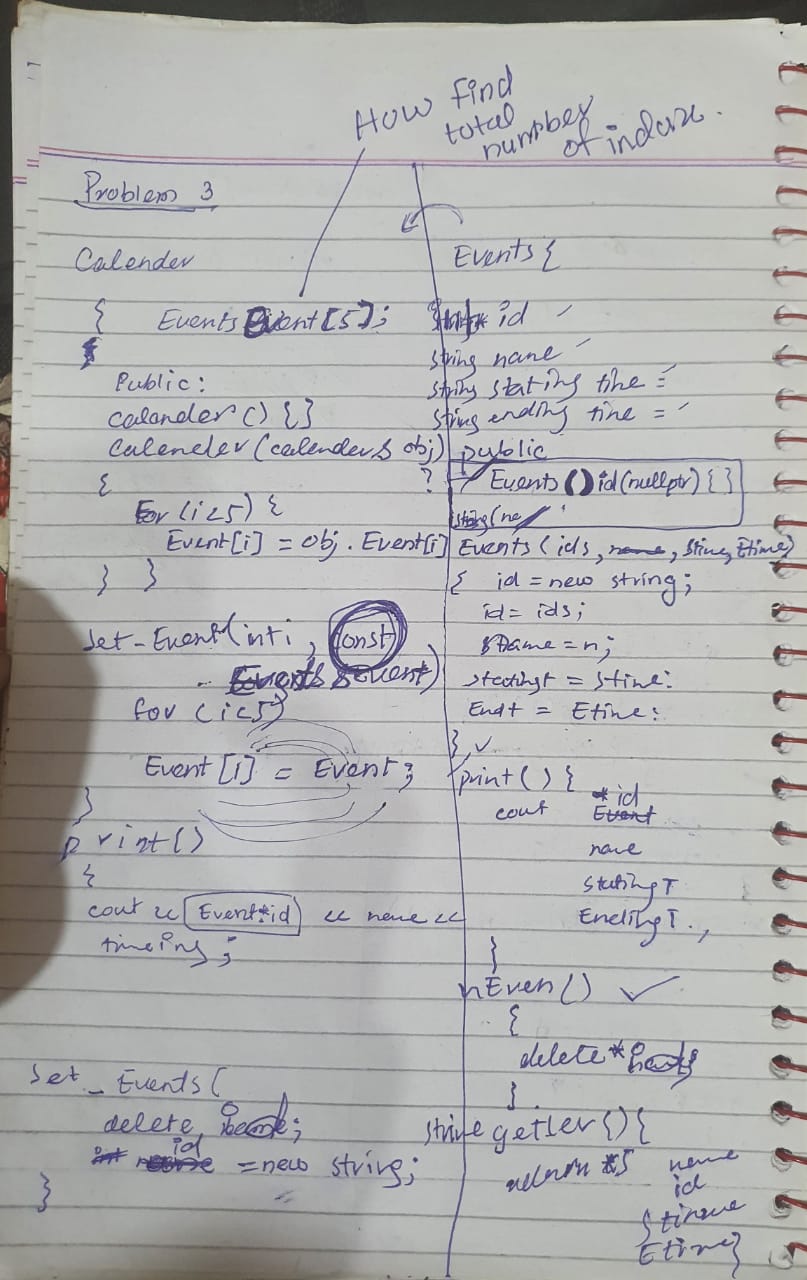
Implement a deep copy constructor for a class that represents a calendar. The Calendar class contains an array of Event, each of event has id, name, start time, and end time. Demonstrate the concept of shallow copy and Deep copy by taking event id as a pointer. Test your implementation by creating two instances of the Calendar class and using the copy constructor to create a deep copy of one of them.

**Source code**

|  |
| --- |
| / Problem Statement 3:  // Implement a deep copy constructor for a class that represents a calendar.  // The Calendar class contains an array of Event, each of event has id, name, start time, and end time.  // Demonstrate the concept of shallow copy and Deep copy by taking event id as a pointer.  // Test your implementation by creating two instances of the Calendar class and  // using the copy constructor to create a deep copy of one of them.  #include <iostream>  using namespace std;  class Event  {  private:  int \*id;  string name;  string starting\_date;  string ending\_date;  public:  Event() : name(""), starting\_date(""), ending\_date("")  {  id = new int;  }  Event(int \*ids, string n, string s\_date, string e\_date)  {  id = new int;  \*id = \*ids;  name = n;  starting\_date = s\_date;  ending\_date = e\_date;  }  ~Event()  {  delete id;  }  Event(const Event &obj)  {  id = new int(\*obj.id);  name = obj.name;  starting\_date = obj.starting\_date;  ending\_date = obj.ending\_date;  }  int get\_id()  {  return \*id;  }  string get\_name()  {  return name;  }  string get\_s()  {  return starting\_date;  }  string get\_e()  {  return ending\_date;  }  };  class Calender  {  Event event[5];  public:  Calender() {}  Calender(Calender &obj)  {  for (int i = 0; i < 5; i++)  {  event[i] = obj.event[i];  }  }  void set\_event(int i, const Event &events)  {  event[i] = events;  }  void display()  {  cout << "---------------------------" << endl;  for (int i = 0; i < 5; i++)  {  cout << "Id : " << event[i].get\_id() << endl;  ;  cout << "Name: " << event[i].get\_name() << endl;  cout << "Starting data: " << event[i].get\_s() << endl;  cout << "Ending data: " << event[i].get\_e() << endl;  cout << "---------------------------" << endl;  }  }  };  int main()  {  Calender cal;  int id1 = 12, id2 = 13, id3 = 54, id4 = 15, id5 = 16;  cal.set\_event(0, Event(&id1, "School function", "22-04-24", "25-04-24"));  cal.set\_event(1, Event(&id2, "College function", "1-04-24", "3-04-24"));  cal.set\_event(2, Event(&id3, "University function", "5-04-24", "6-04-24"));  cal.set\_event(3, Event(&id4, "Marrige function", "3-04-24", "4-04-24"));  cal.set\_event(4, Event(&id5, "Brithday function", "3-04-24", "6-04-24"));  cal.display();  Calender cal2 = cal;  cout << "8888888888888888888888888888888888888888888" << endl;  cout << "Deep copy" << endl;  cout << "8888888888888888888888888888888888888888888" << endl;  cal2.display();  return 0;  } |

**Output**

****

****

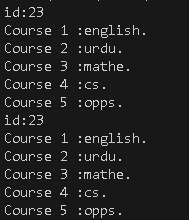
**Problem Statement 4:**

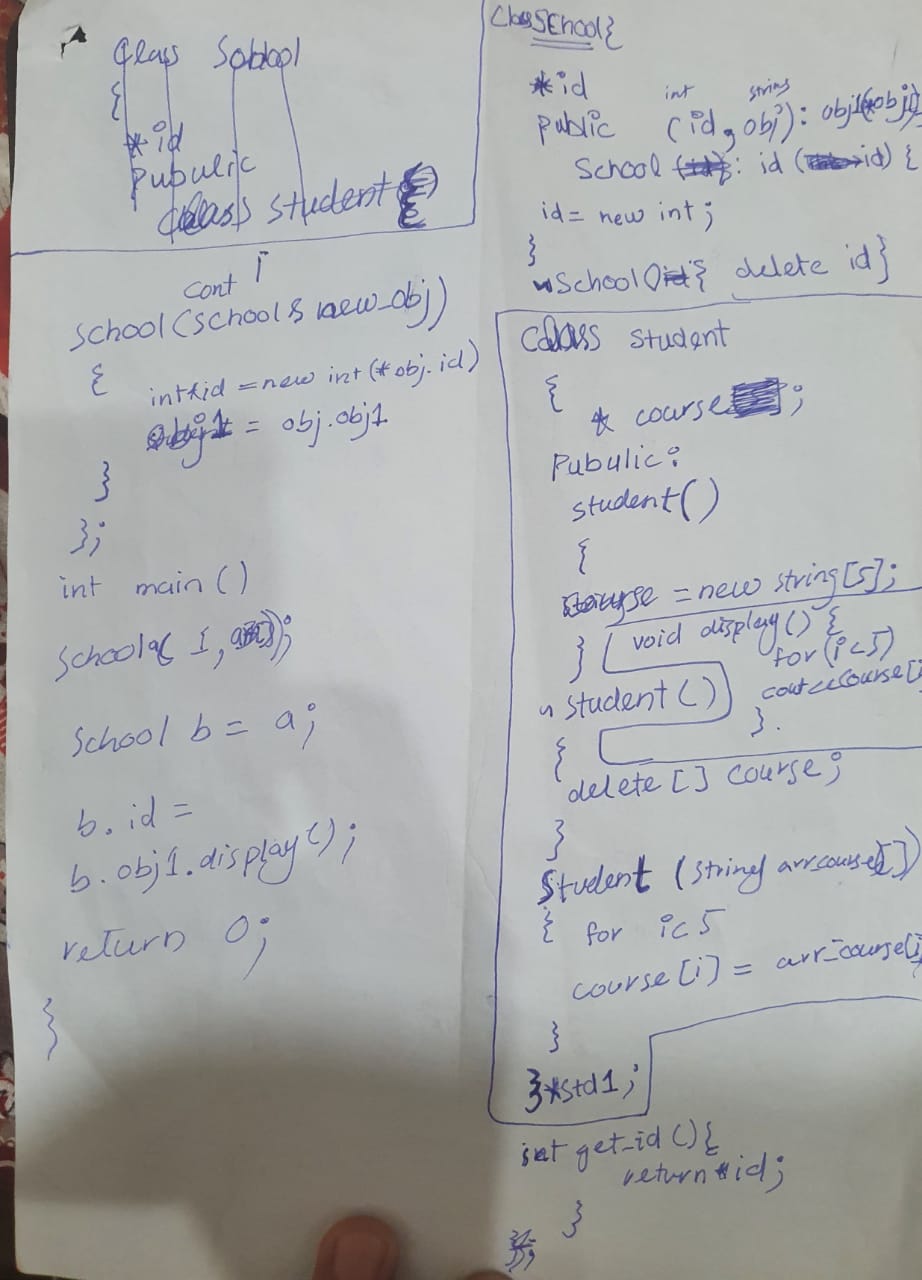
Implement a deep copy constructor for a class that represents a school. The School class contains a pointer to Student objects say school id pointer, which represent the students enrolled in the school. Each Student object has a dynamically allocated array of Course objects, which represent the courses that the student is taking. Test your implementation by creating two instances of the School class and using the copy constructor to create a deep copy of one of them.

**Source code**

|  |
| --- |
| // Implement a deep copy constructor for a class that represents a school.  // The School class contains a pointers to Student objects say school id pointer,  // which represent the students enrolled in the school. Each Student object has a dynamically allocated array of Course objects,  // which represent the courses that the student is taking.  // Test your implementation by creating two instances of the School class and  // using the copy constructor to create a deep copy of one of them.  #include <iostream>  using namespace std;  class School  {          int\* id;      public:          School(int ids, string obj[])          {              id = new int;              \*id = ids;              std1 = new Student();              std1->set\_courses(obj);            }          ~School()          {              delete id;          }          class Student          {                  string \*course;              public:                  Student()                  {                      course = new string[5];                      for (int i = 0; i < 5; i++)                      {                          course[i] = "";                      }                    }                  ~Student()                  {                      delete[] course;                  }                  Student(const Student& other) {                      course = new string[5];                      for (int i = 0; i < 5; i++)                      {                          course[i] = other.course[i];                      }                  }                        void set\_courses(const string obj[])                      {                      for (int i = 0; i < 5; i++)                      {                          course[i] = obj[i];                      }                    }                  void display()                  {                      for (int i = 0; i < 5; i++)                      {                          cout << "Course " << i + 1 << " :" << course[i] << "." << endl;                      }                  }          } \*std1;          int get\_id()          {              return \*id;          }          School(const School &copying\_obj)          {              id = new int(\*copying\_obj.id);              std1 = new Student(\*copying\_obj.std1);          }  };  int main()  {      string courses[5] = { "english", "urdu", "mathe","cs","opps"};      School s1(23,courses);      cout << "id:" << s1.get\_id();      cout << endl;      s1.std1->display();      School s2 = s1;      cout << "id:" << s2.get\_id();      cout << endl;      s2.std1->display();        return 0;  } |

**Output**

****

****