CS250 Class Exercise

Date: Wednesday, March 18, 2009

Complete the following program.

```
// class Point23 represents a point that may be two- or
// three-dimensional, depending on which constructor is used to create
// it. Coordinates are stored in a dynamically allocated array.

class Point23
{
public:
  // default class constructor (with no arguments):
  Point23();

  // class constructor that creates a 2-dimensional point with
  // coordinates xval and yval:
  Point23(int xval, int yval);

  // class constructor that creates a 3-dimensional point with
  // coordinates xval, yval, and zval:
  Point23(int xval, int yval, int zval);

  // class destructor:
  ~Point23();

  // member functions for getting values of x, y, and z (if it's present)
  int Get_X() const;
  int Get_Y() const;
  int Get_Z() const;

// private data members: size of the point and a pointer to the array of
// coordinates:
private:
  int Size;
  int *DataPtr;
};
```

```
// default class constructor creates a 2-dimensional point with
// coordinates x = 0, y = 0.
Point23::Point23()
{
  Size = 2;

  // Exercise 1: allocate memory for an array of two integers









  // Exercise 1: initialize the coordinates to 0s:









}

// class constructor creates a 2-dimensional point with
// coordinates xval, yval.

Point23::Point23(int xval, int yval)
{
  Size = 2;

  // Exercise 1: allocate memory for an array of two integers









  // Exercise 1: initialize the coordinates:









}
```

```
// class constructor creates a 3-dimensional point with
// coordinates xval, yval, and zval

Point23::Point23(int xval, int yval, int zval)
{
  Size = 3;

  // Exercise 1: allocate memory for an array of three integers




  // Exercise 1: initialize the coordinates:




}

// class destructor deallocates all memory allocated for an object of the
// class
Point23::~Point23()
{
  // Exercise 2: deallocate memory for the array of coordinates




}

int Point23::Get_X() const
{
  // every point has an x coordinate
  return DataPtr[0];
}

int Point23::Get_Y() const
{
  // every point has a y coordinate
  return DataPtr[1];
}

int Point23::Get_Z() const
{
  // check if the point has a z coordinate:
  if (Size == 3)
  {
     return DataPtr[2];
  }
```

```
  else
  {
     cout << "attempt to return a third coordinate of a 2D point"
          << endl;
     exit(1);
   }
}

// main: testing the class Point23

int main()
{
// declaring points as variables (statically):
   Point23 p1;            // default: 2D point with coordinates 0, 0
   Point23 p2(3, 4, 5);   // 3D point with coordinates 3, 4, 5

// testing p1, p2:
   cout << "point p1: " << endl;
   cout << "x = " << p1.Get_X() << " y = " << p1.Get_Y() << endl;

   cout << "point p2: " << endl;
   cout << "x = " << p2.Get_X() << " y = " << p2.Get_Y() << " z = " <<
            p2.Get_Z() << endl;

// as an example of allocating memory for an instance of a
// class, we declare a point dynamically:
   Point23 *Ptr1;          // declaring a pointer to a point
   Ptr1 = new Point23(7,8);// allocating memory, initializing the point
                           // to a 2D point with coordinates 7, 8.

// testing the dynamically allocated point:
   cout << "point pointed to by Ptr1 " << endl;
   cout << "x = " << Ptr1->Get_X() << " y = " << Ptr1->Get_Y() << endl;

   // Exercise 3: deleting the dynamically allocated point:


// Note: there is no need to delete p1, p2, since these are local
// variables which are deleted automatically when the program finishes.
// Deleting p1, p2 would be an error, since they have not been allocated
// by "new".

   return 0;
}
```