**Question 1 and Question 4**

#include <iostream>

#include <vector>

using namespace std;

class Point2D

{

public:

Point2D()

{

x = 0;

y = 0;

}

Point2D(double x, double y)

{

this->x = x; this->y = y;

}

double getX() const

{

return x;

}

void setX(double x)

{

this->x = x;

}

double getY() const

{

return y;

}

void setY(double y)

{

this->y = y;

}

// part 1a. function distance()

double distance() const

{

return sqrt(pow(x, 2) + pow(y, 2));

}

private:

double x;

double y;

};

// part 1b. overload insertion(<<) operator

ostream& operator<<(ostream& out, const Point2D& point2d)

{

out << "point( " << point2d.getX() << " , " << point2d.getY() << ")\n";

return out;

}

// part 1c. operator < as nonmember

bool operator<(Point2D p1, Point2D p2)

{

if (p1.distance() < p2.distance())

return true;

else

return false;

}

// Question 4. minPoint()

Point2D minPoint(const vector<Point2D>& v)

{

Point2D smallest = v[0];

for (int i = 0; i < v.size(); i++)

{

if (v[i] < smallest)

smallest = v[i];

}

return smallest;

}

// part 1d. test main

int main()

{

cout << "-----------------------------Question 1-----------------------------\n";

// create two Point2D objects using constructor with parameters

Point2D point1(-6, -2);

Point2D point2(5, 3);

// use overload operator to display points

cout << "point1. " << point1;

cout << "point2. " << point2;

cout << endl;

// call the distance function and display the distance

cout << "The distance from point1 to origin is " << point1.distance() << endl;

cout << "The distance from point2 to origin is " << point2.distance() << endl;

cout << endl;

// compare point1 and point2

cout << "After comparing point1 and point2, ";

if (point1 < point2)

cout << "the smaller object is point1.\n";

else

cout << "the smaller object is point2.\n";

cout << endl;

// Question 4. main

cout << "-----------------------------Question 4-----------------------------\n";

vector<Point2D> vp(5);

vp[0] = point1;

vp[1] = point2;

vp[2] = Point2D(2, 7);

vp[3] = Point2D(-3,-4);

vp[4] = Point2D(1, -9);

// call the distance function and display the distance

cout << "Distance for each element is :\n";

for (int i = 0; i < vp.size(); i++)

cout << vp[i] << "The distance from origin is " << vp[i].distance() << endl;

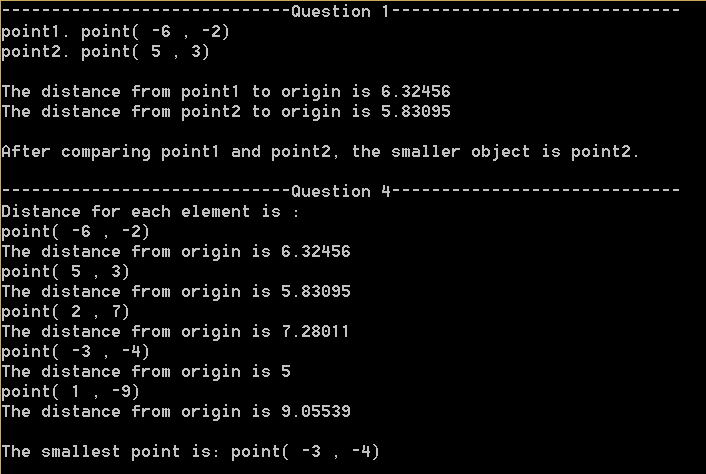
cout << endl;

cout << "The smallest point is: " << minPoint(vp) << endl;

system("pause");

return 0;

}



**Question 2**

#include <iostream>

using namespace std;

class Point2D

{

public:

Point2D()

{

x = 0;

y = 0;

}

Point2D(double x, double y)

{

this->x = x; this->y = y;

}

double getX() const

{

return x;

}

void setX(double x)

{

this->x = x;

}

double getY() const

{

return y;

}

void setY(double y)

{

this->y = y;

}

virtual double distance() const

{

return sqrt(pow(x, 2) + pow(y, 2));

}

private:

double x;

double y;

};

class Point3D : public Point2D

{

public:

// part b. constructors

Point3D() : Point2D()

{

z = 0;

}

Point3D(double x, double y, double z) : Point2D(x, y)

{

this->z = z;

}

// part c. get/set for z

double getZ() const

{

return z;

}

void setZ(double z)

{

this->z = z;

}

// part d. redefine distance function

double distance() const

{

return sqrt(pow(getX(), 2) + pow(getY(), 2) + pow(z,2));

}

// part a. additional private data member

private:

double z;

};

ostream& operator<<(ostream& out, const Point2D& point2d)

{

out << "point( " << point2d.getX() << " , " << point2d.getY() << ")\n";

return out;

}

bool operator<(Point2D p1, Point2D p2)

{

if (p1.distance() < p2.distance())

return true;

else

return false;

}

// part e. test main

int main()

{

// create a Point2D object and a Point3D object using constructor with parameters

Point2D point1(4, 2);

Point3D point2(-1, -2, -3);

// call the distance function and display the distance

cout << "The distance from point1 to origin is " << point1.distance() << endl;

cout << "The distance from point2 to origin is " << point2.distance() << endl;

// part f. polymorphic behavior

Point2D\* ptr;

ptr = &point1;

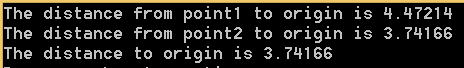
ptr = &point2;

cout << "The distance to origin is " << ptr->distance() << endl;

system("pause");

return 0;

}

****

**The distance function called in part f is from Point3D. The code shows this.**

**Question 3**

#include <iostream>

#include <stdexcept>

using namespace std;

class Point2D

{

public:

Point2D()

{

x = 0;

y = 0;

}

// part a. validating the x- and y- values inside constructor and set functions

Point2D(double x, double y)

{

setX(x);

setY(y);

}

double getX() const

{

return x;

}

void setX(double x)

{

if (x >= -5.0 && x <= 5.0)

this->x = x;

else

throw invalid\_argument("Invalid parameter.\n");

}

double getY() const

{

return y;

}

void setY(double y)

{

if (y >= -5.0 && y <= 5.0)

this->y = y;

else

throw invalid\_argument("Invalid parameter.\n");

}

double distance() const

{

return sqrt(pow(x, 2) + pow(y, 2));

}

private:

double x;

double y;

};

ostream& operator<<(ostream& out, const Point2D& point2d)

{

out << "point( " << point2d.getX() << " , " << point2d.getY() << ")\n";

return out;

}

bool operator<(Point2D p1, Point2D p2)

{

if (p1.distance() < p2.distance())

return true;

else

return false;

}

// part b. exception handling

int main()

{

// create two Point2D objects using constructor with parameters

try

{

Point2D point1(5, 3);

cout << "point1. " << point1;

Point2D point2(-6, -2);

cout << "point2. " << point2;

}

catch (invalid\_argument e)

{

cout << e.what();

}

system("pause");

return 0;

}

****