Old Exam Question

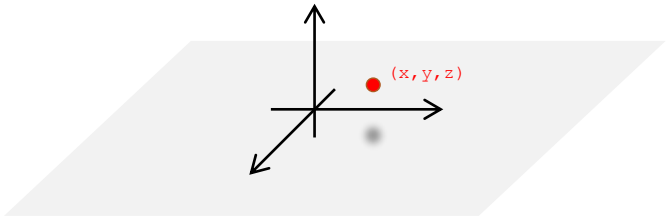
Feb. 2010, Ex. 5

Exercise

In this exercise we will implement a representation of 3D-geometrical objects in a computer game.

Given is a struct `point` which stores 3D-points.

```
struct point {
  double x, y, z;
};
```



(From: Exam Feb. 2010, Ex. 5)

Exercise a)

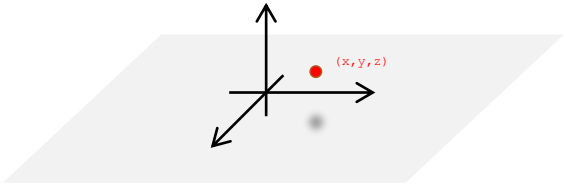
Exercise a)

Implement the following function which computes the distance between a given point and the origin.

Hint: The function `std::sqrt (double d)` computes the square root of `d`.

```
// POST: returns the distance between p and
//       the origin
double distance(const point& p);
```

```
struct point {
  double x, y, z;
};
```



(From: Exam Feb. 2010, Ex. 5)

Exercise a)

Solution a)

```
// POST: returns the distance between p and the origin
double distance(const point& p) {
  return std::sqrt(p.x * p.x + p.y * p.y + p.z * p.z);
}
```

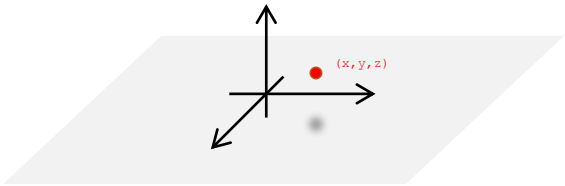
Exercise b)

Exercise b)

Propose a struct named `line`, which can be used to represent 3D-straight-lines.

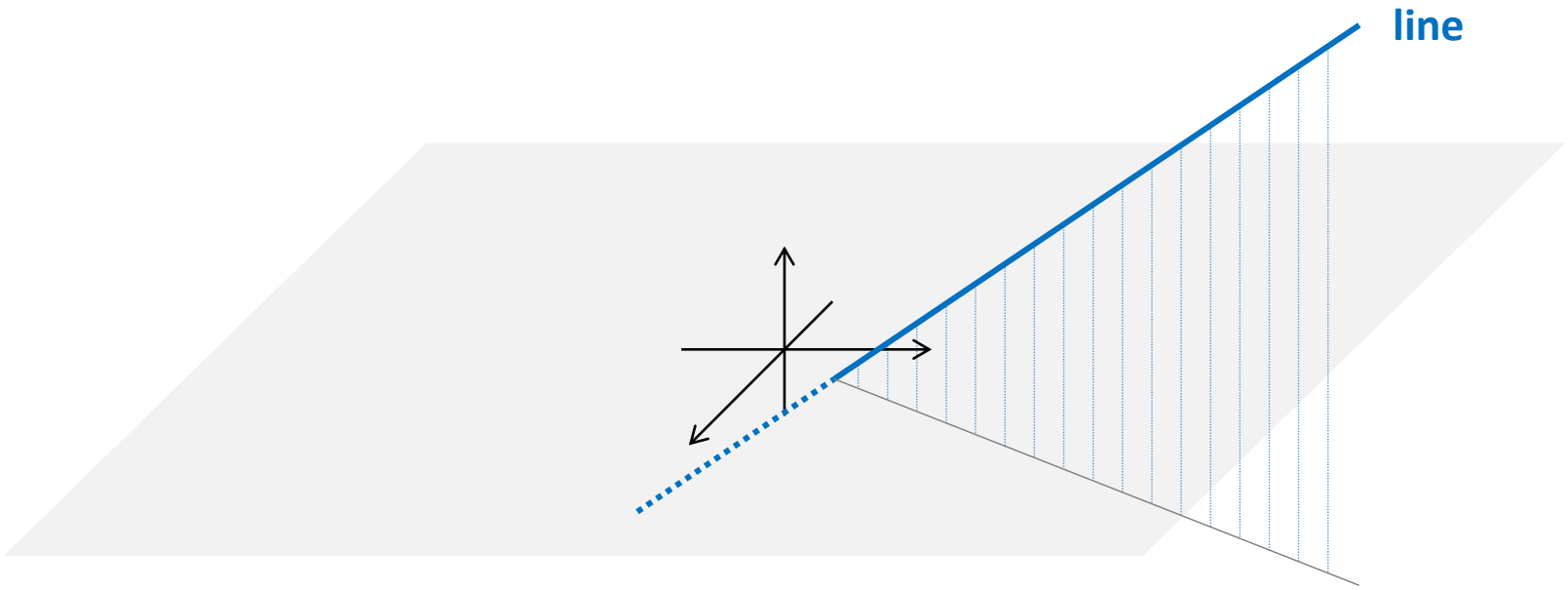
A particular straight line does not have to be representable uniquely, but conversely every object of type `line` has to represent a unique straight line. If necessary you can for this reason define a suitable invariant (`// INV:...`) which has to be met when using the `line` struct.

```
struct point {
  double x, y, z;
};
```



Exercise b)

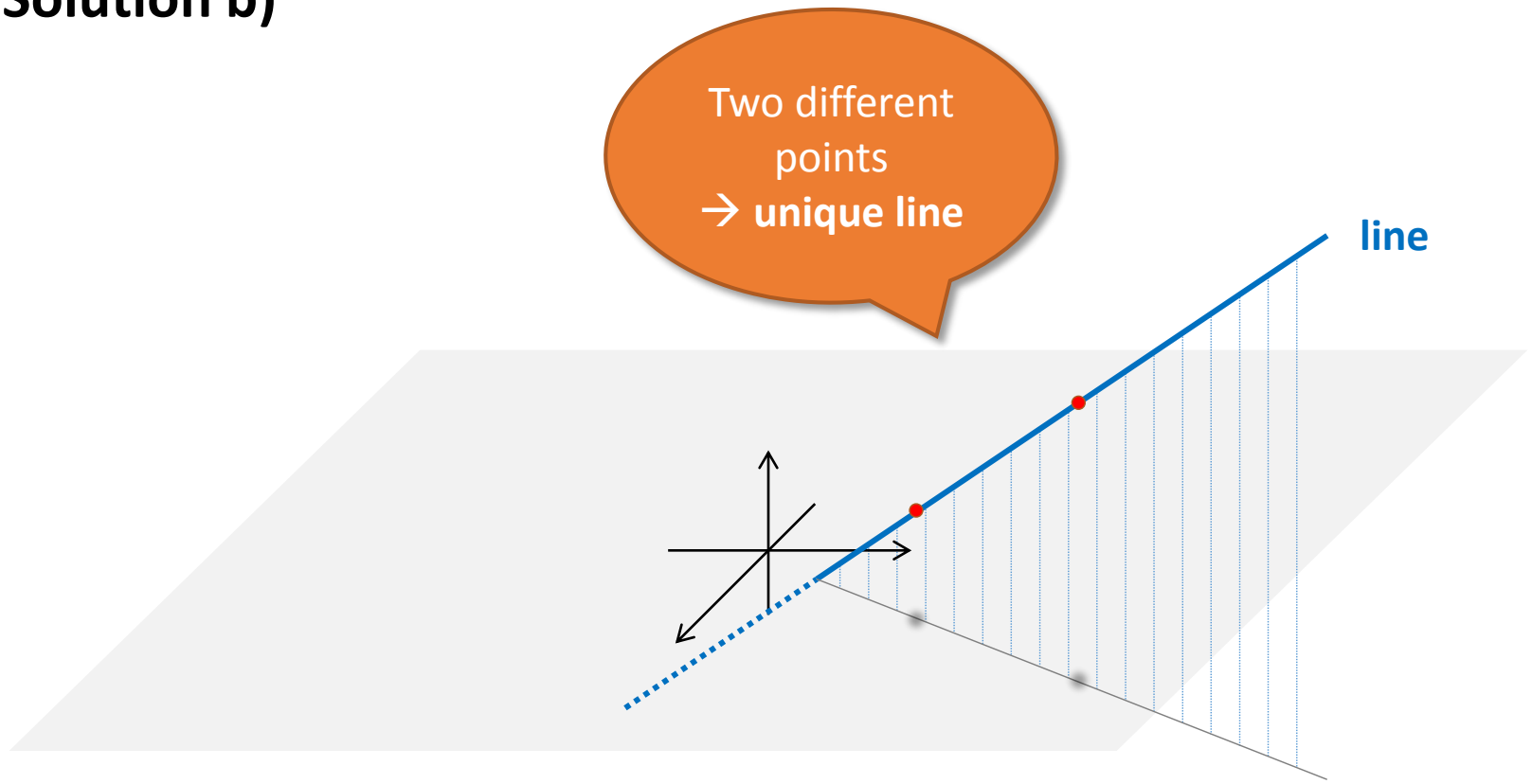
Solution b)



(From: Exam Feb. 2010, Ex. 5)

Exercise b)

Solution b)



(From: Exam Feb. 2010, Ex. 5)

Exercise b)

Solution b)

```
struct line {
  point a, b; // INV: a != b
};
```

Exercise c)

Exercise c)

Based on your struct `line` implement the following function which computes the straight line through two points.

Make sure to meet your invariant from part b). You should define and verify a suitable PRE-condition for this reason.

```
// POST: returns a straight line through a and b
line compute_line (const point& a, const point& b);
```

```
struct point {
  double x, y, z;
};
```

```
struct line {
  point a, b; // INV: a != b
};
```

(From: Exam Feb. 2010, Ex. 5)

Exercise c)

Solution c)

```
// PRE: a != b
// POST: returns a straight line through a and b
line compute_line (const point& a, const point& b) {
  assert( (a.x != b.x) || (a.y != b.y) || (a.z != b.z) );
  line g;
  g.a = a;
  g.b = b;
  return g;
}
```

(From: Exam Feb. 2010, Ex. 5)