2 A computer program is designed to simulate horses doing show jumping. In show jumping, horses jump over obstacles called fences. A horse successfully jumps a fence if it does not knock the fence down.

The program is written using Object-Oriented Programming (OOP).

The class `Horse` stores data about the horses.

Horse	
`Name : STRING`	stores the name given to the horse
`MaxFenceHeight : INTEGER`	stores the maximum height in cm that the horse can jump, for example 132
`PercentageSuccess : INTEGER`	stores the percentage chance of a horse not knocking down a fence, for example 70 represents a 70% chance of jumping a fence successfully
`Constructor()`	initialises `Name`, `MaxFenceHeight` and `PercentageSuccess` to its parameter values
`GetName()`	returns the name of the horse
`GetMaxFenceHeight()`	returns the maximum height the horse can jump
`Success()`	calculates and returns the percentage chance of a horse successfully jumping a specific fence

(a) (i) Write program code to declare the class `Horse` and its constructor.

Do **not** declare the other methods.

Use your programming language's appropriate constructor.

All attributes must be private. If you are writing in Python, include attribute declarations using comments.

Save your program as **Question2_N24**.

Copy and paste the program code into part **2(a)(i)** in the evidence document.

[4]

(ii) The get methods `GetName()` and `GetMaxFenceHeight()` each return the relevant attribute.

Write program code for the get methods.

Save your program.

Copy and paste the program code into part **2(a)(ii)** in the evidence document.

[3]

(b) The array `Horses` stores objects of type `Horse`.

(i) The program has two horses:

- The horse named 'Beauty' can jump a maximum height of 150 cm and has a success percentage rate of 72%.
- The horse named 'Jet' can jump a maximum height of 160 cm and has a success percentage rate of 65%.

Write program code to:

- declare the array, `Horses`, local to the main program with space for **two** `Horse` objects
- store the **two** horses described in the array
- output the name of both `Horse` objects from the array.

Save your program.

Copy and paste the program code into part **2(b)(i)** in the evidence document.

[5]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(b)(ii)** in the evidence document.

[1]

(c) The class `Fence` stores data about the fences. Each fence has a height in cm and a risk number.

The risk is a whole number between 1 and 5 inclusive. A risk of `1` means the fence is the easiest type to jump. A risk of `5` means the fence is the hardest type to jump.

Fence	
`Height : INTEGER`	stores the height of the fence in cm the height is between 70 and 180 inclusive
`Risk : INTEGER`	stores the risk as a whole number between 1 and 5 inclusive
`Constructor()`	initialises `Height` and `Risk` to its parameter values
`GetHeight()`	returns the height of the fence
`GetRisk()`	returns the risk of the fence

(i) Write program code to declare the class `Fence`, its constructor and get methods.

Use your programming language's appropriate constructor.

All attributes must be private.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part **2(c)(i)** in the evidence document.

[4]

(ii) The array `Course` stores **four** `Fence` objects. The user inputs the height and risk for each fence, and these are validated before each fence is created.

Amend the main program to:

- declare the local array `Course`
- take as input the data for **four** fences from the user
- loop the input until both the height and risk are valid for each fence
- create an instance of `Fence` for each of the **four** valid fences and store each instance in the array.

Save your program.

Copy and paste the program code into part **2(c)(ii)** in the evidence document.

[5]

(d) The chance of a horse jumping a fence without knocking it down is calculated as follows.

If the height of the fence is more than the maximum height a horse can jump, the success percentage is 20% of the horse's `PercentageSuccess`. The risk does not affect this value.

If the height of the fence is less than or equal to the maximum height a horse can jump, the risk gives a modifier value to multiply with the horse's `PercentageSuccess`.

The risk values and their modifiers are given in this table:

Risk	Modifier
5	0.6
4	0.7
3	0.8
2	0.9
1	1.0

For example:

- The horse Jet has `PercentageSuccess` of 65 and `MaxFenceHeight` of 160.
- A fence has a height of 140 and a risk of 3.
- The height of the fence is less than the horse's `MaxFenceHeight`, therefore the risk is used.
- The risk of 3 gives the modifier 0.8.
- The modifier 0.8 is multiplied by the horse's `PercentageSuccess` of 65, which gives 52.
- The chance of the horse successfully jumping this fence is 52%.

The method `Success()` in the `Horse` class:

- takes the height and risk of a fence as parameters
- calculates the percentage chance of success for that horse jumping the fence without knocking it down
- returns the calculated percentage chance of success as a real number.

Write program code for `Success()`.

Save your program.

Copy and paste the program code into part **2(d)** in the evidence document.

[5]

(e) (i) Write program code to amend the main program to:

- calculate and output the chance of the first horse jumping each of the **four** fences without knocking each fence down
- calculate and output the chance of the second horse jumping each of the **four** fences without knocking each fence down.

All outputs must have appropriate messages including the name of the horse and the fence number.

An example output for one horse jumping two fences is:

```
"The horse Fox at fence 1 has a 68% chance of success
The horse Fox at fence 2 has a 72% chance of success"
```

Save your program.

Copy and paste the program code into part **2(e)(i)** in the evidence document.

[3]

(ii) Write program code to amend the main program to:

- calculate and output the average chance of success for each horse jumping over all **four** fences without knocking each fence down (the average is the total of values divided by the quantity of values). An example output for one horse jumping all of the fences is:

```
"The horse Fox has an average 70% chance of jumping over all four
fences"
```

- output the name of the horse that has the highest average chance of success.

You can assume that each average will be different.

All outputs must have appropriate messages.

Save your program.

Copy and paste the program code into part **2(e)(ii)** in the evidence document.

[2]

(iii) Test your program with the following input data for **four** fences:

Height	Risk
152	5
121	1
130	3
145	4

Take a screenshot of the output.

Save your program.

Copy and paste the screenshot into part **2(e)(iii)** in the evidence document.

[2]