What​ ​we​ ​expect

In​ ​the​ ​solution​ ​to​ ​this​ ​challenge, ​we​ ​are​ ​looking​ ​for​ ​the​ ​following:

* Simplicity
  + Functions​ ​should​ ​be​ ​small
* Readability
  + Functions​ ​and​ ​variable​ ​names​ ​should​ ​strike​ ​a​ ​balance​ ​between​ ​being​ ​short​ ​and clear
* Testability
  + Include​ ​some unit​ ​tests​ ​for​ ​your​ ​core​ ​logic​ ​code

What​ ​we​ ​do​ ​not​ ​expect

These​ ​are​ ​things​ ​we​ ​do​ ​not​ ​care​ ​about​ ​in​ ​this​ ​solution, ​​where​ ​possible, ​​avoid​ ​spending​ ​time​ ​on these​ ​elements.

* Extensive​ ​use​ ​of​ ​3rd​ ​party​ ​libraries
  + Some​ ​use​ ​of​ ​libraries/nuget​ ​is​ ​normal, ​but​ ​we​ ​don’t​ ​need​ ​to​ ​see​ ​lots​ ​of​ ​dependency injection​ ​or​ ​fancy​ ​log​ ​libraries
* Enterprise​ ​directory​ ​structure
  + This​ ​is​ ​a​ ​small​ ​application, ​​it​ ​is​ ​not​ ​necessary​ ​to​ ​demonstrate​ ​the​ ​layout​ ​needed for​ ​much​ ​larger​ ​applications.

Challenge Dog2Bone

A​ dog ​must​ find the bone walking through​ ​a​ kitchen with sleeping cats. ​​Write​ ​a​ ​program​ ​that​ ​will​ ​read​ ​the​ ​initial​ ​game​ ​settings from​ ​one​ ​file​ ​and​ ​a​ ​sequence​ ​of​ ​moves​ ​from​ ​a​ ​different​ ​file.

Then​ ​the​ ​program​ ​will​ ​output​ ​if​ ​the​ ​sequence​ ​leads​ ​to​ ​success​ ​or​ ​failure (wake up cat, out of bounds) for the dog.

The​ ​program​ ​should​ ​also​ ​handle​ ​the​ ​scenario​ ​where​ ​the​ ​dog​ ​doesn’t​ ​find the bone ​or doesn’t​ ​wake up any cat.

Inputs

The​ ​board​ ​is​ ​a​ ​grid​ ​of​ ​n​ ​by​ ​m​ ​number​ ​of​ ​tiles:

5x4​ ​Board

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

The​ ​starting​ ​position​ ​is​ ​a​ ​tile​ (x,y) ​and​ ​the​ ​initial​ ​direction​ ​of the​ dog​ ​is​ ​facing​ ​(that​ ​is:​ ​north,​ ​east, south,​ ​west):

Starting​ ​position: x​ ​=​ ​0, ​y​ ​=​ ​1, ​dir​ ​=​ ​North

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

The​ ​bone​ ​point​ ​is​ ​a​ ​tile (x,y)

Bone​ ​point: ​x​ ​=​ ​4, ​y​ ​=​ ​2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

The​ ​cats​ ​are​ ​defined​ ​as​ ​a​ ​list​ ​of​ ​tiles​ (x,y)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Dog​ ​actions​ ​can​ ​be​ ​either​ ​a​ ​​move​​ ​(m)​ ​one​ ​tile​ ​forward​ ​or​ ​​rotate​​ ​(r)​ ​90​ ​degrees​ ​to​ ​the​ ​right. For example starting in previous board and with the actions **[m,r,m]**, the new board will be.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Backend

Using C#, develop all the interfaces, classes, unit tests… needed to complete the challenge. The input files are:

* A​ ​game​ ​settings​ ​file​ ​including​ ​board​ ​size, ​start​ ​position, ​bone​ ​position​ ​and​ ​a​ ​list​ ​of​ ​cats​ ​positions. Extra: Several game setting files.
* Moves​’ ​files​ ​containing​ ​a​ ​list​ ​of​ ​moves​ (either​ ‘m’​ ​or​ ‘r’). At least one file per expected result.

The output of the backend section should be a​ ​message​ ​on​ ​the​ ​console ​ ​describing​ ​the​ ​result. Possible results are ​Success, ​Cat​ Awake, ​Still Looking ​​and ​Out Of​ ​Bounds.

Frontend

With a framework of your election (Angular, Razor, React, Vue, Aurelia…), use the previous backend interfaces and classes to create a simple frontend (we’re not looking for a computer game) where:

* The game settings are loaded in a board. If we have several initial settings, we should be able to choose between them to start.
* Any of the moves’ files could be selected to be executed.
* After executing, board is updated and the result appears.

Extra: Execution step by step, so we can see the effect of every move.

Deliverables

Zip containing solution and all the input files. Must be uploaded in a cloud storage (Drive, Dropbox…) and share with a link to download.