Homework

Task 1

- use the **`\Module09-Typical-Problems\HomeworkLeak`** project
- just **`dotnet run -c Release`** it and you should observe XML output being constantly printed
- observe it under the **`dotnet-counters`** (live or by visualizing CVS data). What's growing? Do GCs are happening? Do we have fragmentation? In other words - is this "no GC", "fragmentation" or a "real" memory leak problem?
- (optional) record around few minutes **`gc-collect`** session with **`dotnet-trace`**. What are the reasons for the GCs?
- make a GC heap snapshot with the help of **`dotnet-gcdump`** and yet another after a minute or two. Compare them in your tool of preference - PerfView or Visual Studio (I suggest both, for practice). What's the reason of the memory growth?
- increase in number of object should be clear, but still not so obvious who creates them. Use **`dotnet-trace`** with **`gc-verbose`** profile to see that 👀 That's a spoiler - we will cover allocations in the next module!

(optional, nerdy!) Task 2 - playing with `GCPerfSim`

This is an open exercise without any "solutions". I'd like to invite you to play with the tool that the .NET team is itself using in profiling GC behavior in various scenarios. It's part of dotnet/performance repository about the GC. All the repository is pretty complex and have some prerequisites like Python because of extensive scripting and summary generation.

But in the end, it is running the **`GCPerfSim`** application based on the extensive configuration that simulates various app behaviors. For example, the survival rate of small objects, how big objects should be created etc.

I've extracted it to the homework repository. So to play with it, just build & run it, for example:

```
❯ dotnet run -c Release --framework netcoreapp5.0 -file .\scenarios\sample.test
```

The configuration file may contain many options, refer to the source code for comments. In general, test consists of one or more **phases** executed sequentially. In every phase you can ask for allocating one or multiple **buckets** of objects - specifying their size range, survival rate etc.

(optional, nerdy!) Task 2 - playing with `GCPerfSim`

For example, **`sample.test`** specifies that there should be **`4`** allocating threads, GC stats should be printed every **`1000000`**th iteration and there is only a single phase of **`reference`** type (sample objects that contains references and can form linked list), total live data size should be **`4`**GB and test will end after **`128`**GB of allocations. And there is a single bucket of objects with size between **`100`**-**`4000`** bytes, where every **`30`**th object will survive and there is no pinning.

```
threadCount 4
printEveryNthIter 1000000

[phase]
allocType reference
testKind time
totalLiveGB 4
totalAllocGB 128

[bucket]
lowSize 100
highSize 4000
survInterval 30
pinInterval 0
weight 1
```

So... play with various configs, try to observe interesting behavior, observe various tests under **`dotnet-trace`** or **`dotnet-counters`**. YMMV!