(this started out as a normal .txt file, but then I realised I had a nice way of formatting it...)

DISCLAIMER

The last time I programmed anything in C# was around 2 years ago for another internship coding test so I was pretty rusty while doing this. Also I ended up only putting half the project on git(https://github.com/NoNotCar/Gearset-PDF) after finding out that tests don't go in the main project folder (which I'd put on git) and not wanting to mess around with the project structure a bunch and corrupting something by accident.

RUNNING

Program.cs is currently set up to use repeated.txt as input and output to output.pdf

The development process

So I started off the project using Visual Studio Code because I already had it installed and had good experiences with both javascript and python in it. I then realised the limitations of the IDE about halfway through when I wanted to find the font class in iText and desperately needed suggestions that the IDE couldn't provide.

After switching to Visual Studio, I finally managed to actually add a separate tests project and then promptly realised that the structure of having one thing parsing and writing the document was quite difficult to write tests for. I then changed the structure to the current structure, which would theoretically allow writing to a different format (e.g. HTML) fairly easily. However, I still couldn't think of good ways to unit test it (I'd still need to mock a bunch of stuff, which seemed like it'd take a while to set up properly) and so decided to use snapshot tests instead. I briefly looked at available packages for this, but ended up writing my own implementation as it turns out PDFs store some kind of unique ID and creation date internally which I needed to ignore when comparing them. Overall I think the snapshot system works pretty well as it gives a good visual indication of what each test is doing, however certain changes (such as changing the default font size) would require rewriting all the snapshots.

Assumptions made:

- .large and .normal are commands, despite not appearing in the list of commands

- text gets wrapped but not justified by default (I don't see why you *wouldn't* want text wrap, and you can use .paragraph to force new lines)

- indentation can't go below zero

- unknown commands should be ignored (and print warning in the console), likewise for unparseable indentation

Extensions I don't have time for

- actual unit tests (yeah, if I'd been doing test-driven development from the start I'd have run into far less issues...)

- less repetitive snapshot tests (Putting the assertion in the SnapshotTest method (if that's allowed) or using [Theory] or something if that will show up as multiple tests)

- putting EVERYTHING on git

- some kind of alternative to the monster switch statement in InputParser?

- more text size commands/.size [SIZE] command

- using something other than iText because the online documentation (well for C#, the java version actually looks ok) is spectacularly unhelpful

- interpreting - and * as bullet points because putting .paragraph after all your list items is rather repetitive...

- The HTML writer mentioned earlier, especially as SetBold/SetItalic would actually do something other than be a dumb wrapper for bold/italic (e.g. add a <b> tag to the text)