Part 1

* Assume that you search a keyword in Google and then click a link that goes to a linkedin post
* You’re not logged in Linkedin.
* You can use this post link as a sample: <https://www.linkedin.com/posts/austin-eovito-245220152_lets-talk-about-artificial-intelligence-activity-7171294281360470016-T0iC>
* (I searched “artificial intelligence” keyword in Google to get this linkedin post link)
* After clicking this link while you’re not logged in linkedin, you reach the linkedin post page.
* Save as HTML that page into a text file.
* Parse this HTML file so that we can extract **author username, author name surname, author title, content of the post, how many likes/republish/comments ithas, publish date**
* Return these extracted values as JSON
* You can use beautifulsoup Python library to parse HTML content.
* You dont need to simulate Google search to find linkedin post links.
* You just have HTML content of a linkedin post and we need to parse this file.

Part 2

* 1. You can divide code part and html separators. Read the html separators as variables from a config file. (for example, like\_count\_separator= "class\_='font-normal ml-0.5'")
* 2. You assume that you'll have a value after parsing a specific separator from a html source.  
  It is better to control if that value is not null and then assign it the returned the value
* 3. If the returned value is null, integer variables should be set to 0 in JSON, string variables should be set to ""
* 4. Republish count should be added (in this sample it was not republished, but we should implement it. You can test it with a different post that has been republished)
* 5. Author username should be added
* 6. You can set variable names as singular. (comments\_count -> comment\_count)
* 7. Research on publish date whether we can get it as a regular date format: 21/03/2024 from the HTML source. (check if it is available in html source)

Part 3

* 1. some part of the code may throw an exception when unexpected data is received. So I want to you add "try except" column to your code
* 2. Extract your code to methods  
   a) The methods take html content as parameter  
   b) Return the result with type of dictionary
* 3. Test your code with more and different html files.

Part 4

* 1. There is a bug in the code that needs to be fixed. If you test the code with different html files you will notice the bug.
* 2. Convert date of publishing localdatetime. Example not like "**1w**" but it must be like "**2024-03-26T09:52:56.366447**"
* 3. Extract from the html content member id. The id is in the "<a>" tag with data-semaphore-content-urn="urn:li:activity:7171294281360470016".

Part 5

1. You can take user profile link and username from the html source.
2. Extract methods to class
3. Create another class which is model dataclass for the structure of the result.
4. And take the parsed time into "**retrieval\_time**" variable and add this to result
5. If member\_id contains only number assign as a "**public\_member\_id"**if member id start with "**ADo**" assign as a "**public\_linkedin\_member\_id"**. then add this to result
6. Get pageLang from html source and assign as a "**page\_lang"**and add this to result