1.

a.https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q5686&format=json
This URL has action: wbgetentities
Id of the search result:Q5686
format:json
The query is everything behind the question mark so that includes action=wbgetentities, ids=Q5686, format=json

b.https://www.wikidata.org/w/api.php?action=help.
For this URL the query parameters are:
action=help

c.https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q5686
Query parameters:
action=wbgetentities
ids=Q5686

2.
GetResultsFromURL(): gives us all the results when searching for last name “dickens” as a list
GetResultsFromURL()[0] : gives us first search result
GetResultsFromURL()[0].id : gives us the id
Then we encode the ID in the URL for the wikidata url which gives us the following URL:
https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q5686
We go to this URL and access the image by doing:
response.claims.P18.mainsnak.value
If response.claims.P18.mainsnak.value gives us null the image does not exist
If it doesn’t give us null the image does exist.

3.
This URL:
https://www.wikidata.org/w/api.php?action=wbsearchentities&language=en&search=dickens
Gives us the ID and it uses one search query

The second URL:
https://www.wikidata.org/w/api.php?action=wbgetentities&ids=Q5686
Where we search for the image uses another search query

So for every image we search for we use two search queries.

Therefore, if the website allows max 1000 requests per day you can only run algorithm 1000/2=500 times

4.
I would have a like button for every wikidata entry and everyone who found the wikidata entry helpful could like it and filter based on how many likes each wikidata entry got.

Targets Completed:
Target1:

- (2) A. A minimal solution looks just like the title pages in the starter code.
    - Extract the author for now by taking the creator field up to the first comma: `Dickens, Charles, 1812-1870` -> `Dickens`
- (2) B. A satisfactory solution has the P2 paging system integrated in the results.
- (3) C. A good system can sort these pages by title, author, and popularity (per the video).
- (5) D. An excellent system extracts the dates from author fields (birth date and death date) if possible. Add sorting by both dates (user's choice).
- (2) E. A better solution keeps authors with the same last name (but different first names) separate, as possible (the identity of the Author class is based on first and last name).
- (4) G. A great system has two-levels of pages, because there is now an Author class.
    - At the first level, links to each author are available, with a count of books by each.
    - At the second level, books by that author are available.
- (4) F. A great system extracts authors from the title (e.g. after the word by) if not available in the creator field.
- (5) H. An excellent system has a more sophisticated solution for books with multiple authors (e.g. etext7010).

Basically what I did for this was see how many authors were in the creator field. A pretty good way to see how many authors were in the creator field were to see how many dates were there. To see how many dates were there I counted the occurence of “-” because each date had one. However some authors naturally had a “-” in their name so I made extra careful to not include those authors by checking to see if the was a letter preceding the “-” or a number. If it’s a number we would include it.

After extracting the author I found that due to the way the creator string is formatted some authors last name had a date attached to it. So I looped through the string and got the ascii of every character and only kept it if it was a string not a number or “-”.

This target was harder than expected and is some of the most intense string processing I have done in one night.

Target 2:

- (2) I. A minimal system tracks likes globally. Every click increments a counter, which is stored per-book and visible to us.
- (2) J. A satisfactory system tracks likes per-user, but has a user field in every book's form in order to like.
- (8) K. A good system will allow users to log in in one place, tracking their username in cookies. Attempting to "like" something without being logged in will actually send the user to the login form. (this still remains on the same page)
- (4) L. A great system can show which users have liked a book.
- (4) M. A great system will also allow a user to visit a "/user/likes" page which shows them what they (or other users) have liked. (Instead of “/user/likes” just go to “/like” to see which books have been liked and by who)
- (4) N. A great system will not lose user likes when the server is restarted.