1 Process queries

Our goal is to write the process_query function in the homework. This is a complicated function so we will build three versions of it, each one progressively more complex.

The inputs to process_query are:

- a string representing a query, such as 'cat dog'
- a reverse index, which is a dictionary whose key-value pairs are a word (the key) and the list of web pages containing that word (the value). Here is an example:

  ```python
  reverse_index = {'cat': ['fake1.html', 'fake3.html'],
                  'dog': ['fake1.html', 'fake2.html'],
                  'horse': ['fake1.html']}
  ```

1. process_query depends on some list processing functions. Write the functions list_union, list_intersection, and list_difference. (Solutions to these functions will be briefly reviewed in class, but will not be posted in the handout.)

   **Challenge Edition** (optional): If you want an extra challenge, try writing list_intersection without using the in operator or the count method, both of which are slow if the lists are big. Instead, sort each list and then use a while loop and move up each list looking for items that occur in both lists. Hint: maintain index i for L1 and j for L2 and increment one, the other, or sometimes both depending on whether L1[i] is smaller, bigger, or equal to L2[j].

   **Solution:** Solutions were reviewed in class.

2. Write a function process_query1 that takes in a query and a reverse index and returns a list of web pages that contain any of the query terms. For example, 'cat dog' should return ['fake1.html', 'fake2.html', 'fake3.html']. You may assume the query contains only lowercase words having only alphabetical characters.

   **Solution:**
   ```python
   def process_query1(query, index):
       terms = query.split()
       matches = []
       for term in terms:
           term_matches = get_query_hits(term, index)
           matches = list_union(matches, term_matches)

       return matches
   ```
3. Write a function `process_query2` that takes in a query and a reverse index and returns a list of web pages that contain any of the query terms, unless the first term is 'AND'. In this case, it should return the web pages that contain all of the query terms. For example, 'AND cat dog' should return ['fake1.html']. You may assume the query contains only lowercase words having only alphabetical characters (except of course the first word may be 'AND').

**Solution:**
```python
def process_query2(query, index):
    terms = query.split()
    if terms[0] == 'AND':
        terms.pop(0)  # remove the AND
        intersect = True
    else:
        intersect = False

    matches = get_query_hits(terms[0], index)  # if intersecting,
    # cannot start with
    # empty matches!

    for term in terms[1:]:
        term_matches = get_query_hits(term, index)
        if intersect:
            matches = list_intersection(matches, term_matches)
        else:
            matches = list_union(matches, term_matches)
    return matches
```

4. Write a function `process_query3` that works like `process_query2` except that it handles the case when the query contains minus terms. A minus term such as '−horse' should be used to filter the results: any web page that contains this term should be removed from the results. Again, you may assume the query contains only lowercase words having only alphabetical characters (except of course the first word may be 'AND').

For example, 'cat dog −horse' should return ['fake2.html', 'fake3.html'].

Another example, 'AND cat dog −horse' should return [].

**Solution:**
```python
def remove_negatives(L):
    '''(list of str) -> list of str
    Removes items from L that start with '−'.
    Returns a list containing the items that have been
    removed. The '−' is removed from each item.
    '''
```
i = 0
exclude_L = []
while i < len(L):
    if L[i].startswith('-'):
        term = L.pop(i)
        exclude_L.append(term[1:])    # remove minus
    else:
        i += 1
return exclude_L

def process_query3(query, index):
terms = query.split()
if terms[0] == 'AND':
terms.pop(0)    # remove the AND
intersect = True
else:
    intersect = False

exclude_terms = remove_negatives(terms)
makes = get_query_hits(terms[0], index)    # if intersecting,
    # cannot start with
    # empty makes!

for term in terms[1:]:
term_matches = get_query_hits(term, index)
if intersect:
makes = list_intersection(makes, term_matches)
else:
makes = list_union(makes, term_matches)

for term in exclude_terms:
term_matches = get_query_hits(term, index)
makes = list_difference(makes, term_matches)

return makes

5. Write the final version of process_query. It is similar to version 3 above. However, each term in
the query must be normalized using the normalize_word function. Read through the homework
to make sure all requirements are handled. Also, test your code using the test cases in the
homework description.