```java
public static void populateTitles() {
    for (String movie : ID_MAP.keySet()) {
        IDeque<String> suffixdeque = new ArrayDeque<>();
        IDeque<String> movieTitles = new ArrayDeque<>(); //initialize new dequeues to map for each loop
        movieTitles.addBack(movie);
        String[] movielist; //initialize an array for the words in the title
        movielist = movie.split( regex: " "); //add a space so we can get the substring
        for (String movies : movielist) { //add each word of the title to the suffixdequeue
            suffixdeque.add(movies);
        }
        for (int i=0; i < movielist.length; i++){
            if (!titles.containsKey(suffixdeque)) { //check to see if our titles has this key (cant have duplicate keys)
                IDeque<String> movieTitles2 = new ArrayDeque<>();
                movieTitles2.add(movie);
                titles.put(suffixdeque, movieTitles2); //if it doesn't, put the mapping in
            } else {
                    titles.get(suffixdeque).add(movie);
            }
            suffixdeque.removeFront(); //always remove the first term to check another suffix
            }
        }
    }
2 usages  blank *
public static IDeque<String> complete(String term) {
    IDeque<String> termDeque = new ArrayDeque<>(); //put the given term into a deque, so we can analyze it
    String[] terms = term.split( regex: " ");
    for (String words : terms){
        termDeque.add(words);
    }
    ICollection<IDeque<String>> completeCollection = new ArrayDeque<>();
    completeCollection = titles.getCompletions(termDeque); //receive all the completions for the passed term
    IDeque<String> completionDeque = new ArrayDeque<>(); //the deque we want to return
    for (IDeque<String> completion : completeCollection) { //loop through all the completion deques in the collection we have
        for (String actualCompletion : completion) { //nested loop to get each actual completion
            if (!completionDeque.contains(actualCompletion))
                completionDeque.add(actualCompletion); //add the completions to the return deque
        }
    }
    return completionDeque;
```