Wendi Tan
wtan37@ucsc.edu

Assignment 6

Pre-lab:

Part1:

```
1). bf_delete(**bf){
        free((*bf)->filter)
        free(*bf)
        *bf = NULL
}

2). bf_insert(*bf, *oldspeak){
        index1 = hash(salt1, oldspeak)
        index2 = hash(salt2, oldspeak)
        index3 = hash(salt3, oldspeak)
        bv_set_bit(bf->filter, index1)
        bv_set_bit(bf->filter, index2)
        bv_set_bit(bf->filter, index3)
}
```

Part2:

```
1). ll_create(bool mtf){
        LinkList *ll = malloc(sizeof(LinkList))
        ll->length = 0
        ll->head = node_create(NULL, NULL)
        ll->tail = node_create(NULL, NULL)
        ll->head->next = ll->tail
        ll->tail->prev = ll->head
        ll->mtf = mtf
    }

2). ll_delete(*ll){
        Node *index = node_create()
        Index = ll->head
        while(loop ends when index == ll->tail){
                node_delete(index)
                index = index->next
        }
        node_delete(ll->tail)
        node_delete(index)
        free(ll)
        ll=NULL
    }

3). ll_length(*ll){
        Length = 0
        Node *index = node_create()
        Index = ll->head->next
        while(loop ends when index == ll->tail){
                length ++
                index = index->next
        }
        Node_delete(index)
        Length = ll->length
}




4). ll_lookup(*ll, char *oldspeak){
        Node *index = node_create()
        index = ll->head->next
        While(index->oldspeak != oldspeak){
                index = index->next
                if(index == ll->tail){
                        Return NULL
                }
        }
        if( index->mtf ){                       // move-to-front operation; *n = index->prev
                n->next = index->next
                index->next->prev = index->prev
                index->next = ll->head->next
                index->prev = ll->head
                ll->head->next->prev = index
                ll->head->next = index
        }
        return index
}

5). ll_insert(*ll, *oldspeak, *newspeak){
        Node *n = node_create(oldspeak, newspeak)
        n->prev = ll->head
        n->next = ll->head->next
        ll->head->next->prev = n
        ll->head->next = n
}

6). ll_print(*ll){
        *index = node_create()
        index = ll->head->next
        while(index == ll->tail){
                node_print(index)
                index = index->next
        }
        node_delete(index)
}
```

Part3:

```
/ (?=\S*['-])([a-zA-Z'-]+)/
```