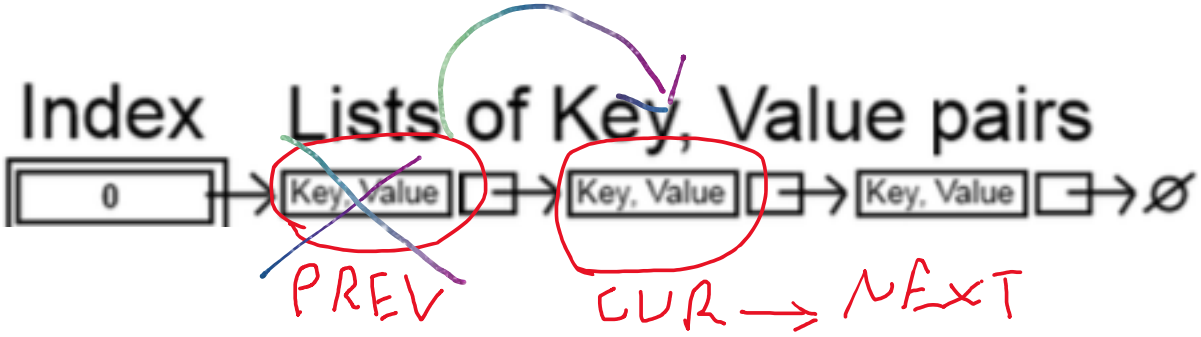
LAB5

Funzione search: Semplicemente il suo compito è quello di scorrere la lista, per farlo uso cur impostato a s[h(k)].
All'interno del ciclo controllo se k (chiave passata dalla funzione) è diversa dalla chiave di cur, in caso positivo restutuisco il valore ed ovviamente questo causa un errore in quanto ricordiamoci che due chiavi uguali NON devono dare lo stesso valore, ma due chiavi diverese POSSONO dare lo stesso valore, per il primo caso si parla di duplicati mentre nel secondo di collisione.

Funzione InsertElement: Il primo controllo che effettuo è quello di verificare se ho chiavi duplicati, lo faccio tramite la chiamata a search.
Mi dichiaro una nuova cella e gli assegno i suoi parametri key e value.
Se s[h(k)] è un bucket vuoto allora semplicemente s[h(k)] diventa aux e il next di aux punta al nulla, in caso contrario aux -> next punta al next di s[h(k)] (sarebbe nullptr perche faccio inserimento in coda) e infine assegno aux ad s[h(k)].

Funzione deleteElem: il ragionamento e simile come facevamo nelle liste semplici.
Quello che vado a fare e avere due puntatori ad s[h(k)].
Io li ho chiamati prev e cur.
Scorro i vari bucket e se trovo l'elemento vado a verificare il caso in testa oppure in mezzo/fondo.
Caso in testa: se cur == prev vuol dire che l'elemento da eliminare è il primo, vado a spostatore il puntatore cur (cur = cur -> next) elimino prev e imposto s[h(k)] a cur, il ragionamento è come nelle liste semplice, la mia lista (s[h(k)]) deve diventare il prossimo elemento (cur).



Mentre per gli altri due casi semplicemente faccio prev -> next = cur -> next ed elimino cur.