NO:

DATE:

- DICTIONARY -

*Dictionary

- list of pointers; elements are grouped by something common (i.e. last digit of num. = index of array)

* Operations

1.) insert () - adds x to A if $x \notin A$

2.) delete () - removes x from A if $x \in A$

3.) member () - returns 0 if $x \notin A$
1 if $x \in A$

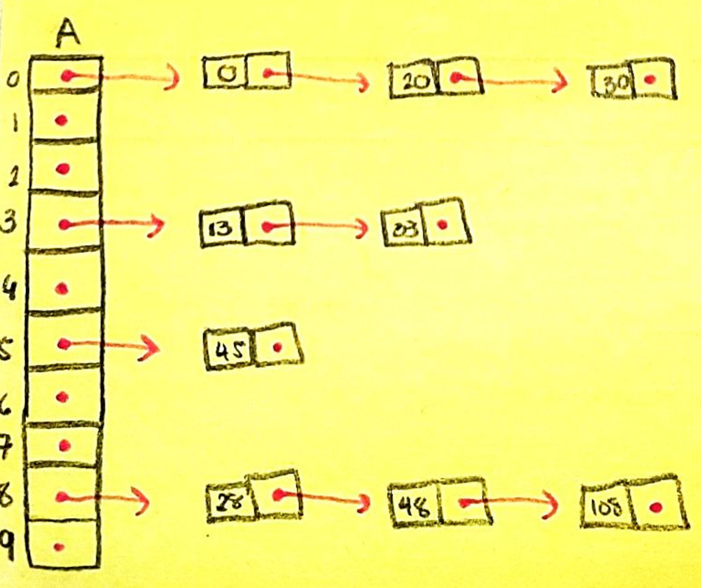
4.) init() - initializes dictionary pointers to NULL

5.) makeNULL - makes dictionary list empty

* Implementations

1.) Linked List	- O(N)
2.) Array	- O(N)
3.) Cursor-based	- O(N)
4.) Hashing	- O(1) improved!

Set A = {0, 13, 20, 28, 30, 33, 45, 48, 108}



*elements are grouped by last digit = index of array. this is easier to ~~traverse to look~~ for an element now bcs its all organized.

* Hashing

- uses function: hash ()
- assigns a "hash value" to an element
- determines:
  > exact location of element (Closed Hashing)
  > starting point in searching (Open Hashing) for element

* Kinds of Hashing

1.) Open Hashing (External)
- allows set to be stored in potentially unlimited space
- array of Linked Lists

2.) Closed Hashing (Internal)
- uses a fixed space for storage & thus limits the size of set
- Array

* Open Hashing

Data Structure:

> array of sets (or groups)
> each set can be array OR linked list

* uses % modulo to get hash value

↳ based on size of

* Hash() function

- returns an INTEGER value = the SUBSET (group) in which the element is a member of

Examples:

- Group integer elements according to ONES digit
- Group names according to 1st letter of name

//hash()

① code of hash fn() that accepts integer as parameter & returns digit in ones place.

```
int hash (int x){
    return (x% MAX);
}
```

②

one's digit		hash value
0,1	→	0
2,3	→	1
4,5	→	2
6,7	→	3
8,9	→	4

```
int hash (int x) {
  return (x%10)/2;
}
```

③ code of hash fn() that accepts lastname as parameter & returns:

DATE:

0 if the 1st letter is A

1 if the 1st letter is B

...

25 if the 1st letter is 25

```
int hash (char lname[]){
    return lname[0] -`A';
}
```

// Assignment

1.) Hash fn() accepts parameter integer x & returns digit in hundreds place of x.

$\underset{\uparrow}{1}00$

```
int hash (int x) {
    return (x/100) % 10;
}
```

2.) Hash fn() that accepts parameter integer x & returns hash value between 0 to 18 (remainder when the sum of all digits is divided by 19)

3.) Hash fn() accepts name & returns HV between 0 to 48. Its remainder when sum of ASCII values of letters in name is divided by 49.

// Assignment

1.) Hash fn() accepts parameter integer x & returns digit in hundreds place of x.

$\underline{1}\underline{0}\underline{0}$
$\uparrow$

```
int hash (int x) {
    return ( x/100) % 10;
}
```

```
② int hash (int x) {
    int temp, retval;

    while (x != 0){
        temp = x % 10;
        x = x / 10;
        retval = retval + temp;
    }
    return (retval % 19);
}
```

```
t hash ( char name []) {
int retval, x, length = strlen (name);

for (x=0; x< length ; x++) {
    retval = retval + name [x];
}

return(retval % 49);
```

NO:

DATE:

// initDictionary ()

passing an array (ptr to 1st elem)

```
void initDictionary(Dictionary D){
    int x;
  for (x=0; x<SIZE; x++){
      D[x]= NULL;
  }
}
```

```
typedef struct node {
      int data;
      struct node *link;
 }*LIST;

typedef LIST Dictionary [SIZE];
```

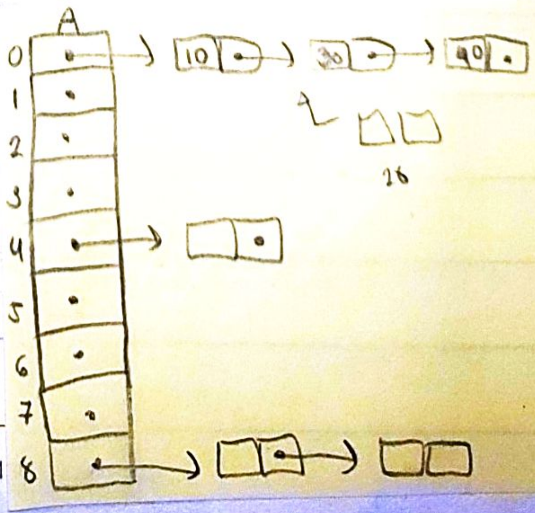
//insert()

```
void insertD(Dictionary D, int x){
     int hashVal= hash(x);  LIST temp=(LIST) malloc(sizeof(sn));
     LIST *trav= &D[hashval];

     for( ;(*trav)!=NULL && (*trav)->data <= x; trav = &(*trav)->link){}

          temp->data=x;
          temp->link = *trav;
          *trav=temp;

  }
```



// delete ()

```
void deleteDictionary (Dictionary D, int x) {
    int hashval = hash(x);
    LIST temp;
    LIST *trav;

    if (D[hashval]! = NULL){
        for(trav= &D[hashval]; (*trav)!=NULL &&
            (*trav)->data != x; trav = &(*trav)->link){}

        temp = *trav;
        *trav = temp -> link;
        free (temp);
    }
}
```

//member ()

```
int isMember (Dictionary D, int x) {
    int hashval = hash (x);
    LIST trav;

    for(trav= D[hashval]; trav != NULL && trav->data
        != x ; trav = trav = trav->link){}

    return (trav != NULL)? 1:0;
}
```