Data Structures and its Applications

V R BADRI PRASAD
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Hashing :

- Insert Operation
- Display Operation

V R BADRI PRASAD

Department of Computer Science & Engineering

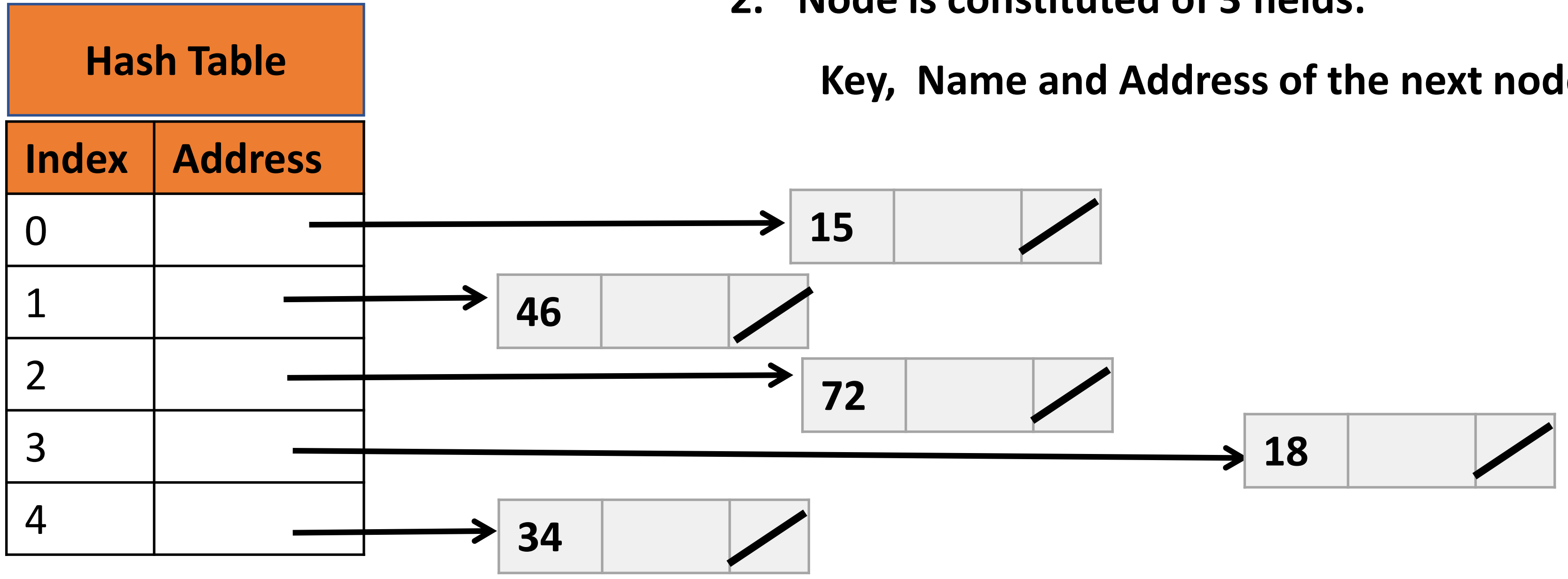
Data Structures and its Applications

Hashing – Open Addressing / Separate Chaining

- Consider key elements as 34, 46, 72, 15, 18
- Hash function is **key mod 5.**

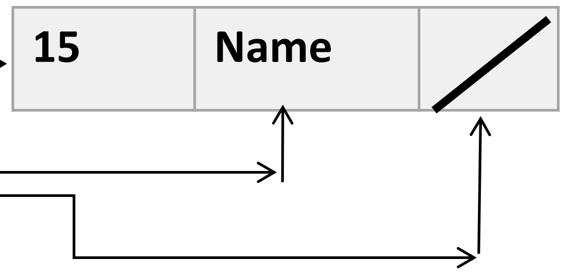
1. **Hash Table contains Index, Address fields.**
2. **Node is constituted of 3 fields:**

   Key, Name and Address of the next node.

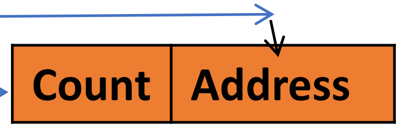


Hashing : Node Creation – Separate Chaining

```
struct node
{
  int key;
  char name[100];
  struct node *next;
};
```



```
struct hash
{
  struct node *head;
  int count;
};
```



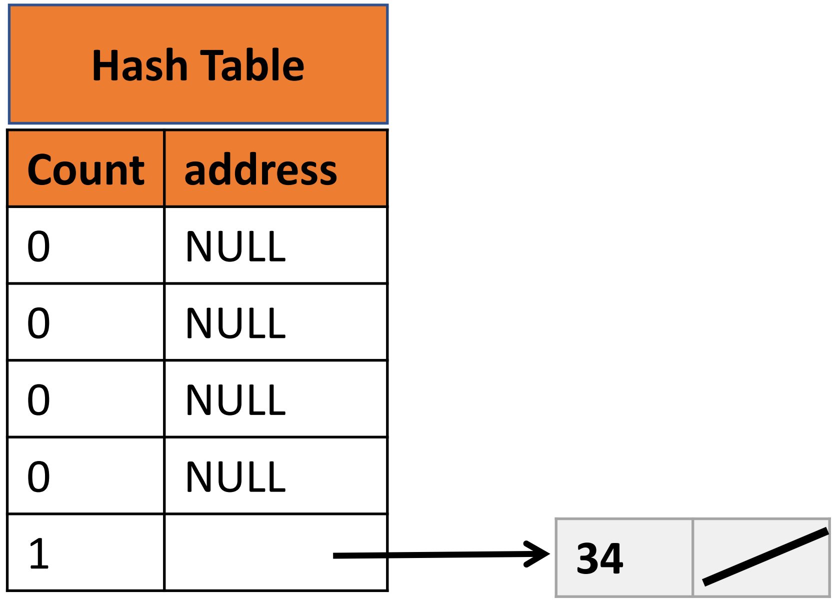
Hashing: Insert Operation

```
void insert_to_hash(struct hash *ht, int size, int key, char* name)
 {
  int index;
  struct node *temp;

//   Create a node and store the starting address in temp variable.

  temp=(struct node*)(malloc(sizeof(struct node)));

  temp->key=key;

  strcpy(temp->name,name);

  temp->next=NULL;
```

Hashing – Open Addressing / Separate Chaining

// Insert node at the beginning of Singly linked list as shown in the figure.

```
 index=key%size;
  temp->next=ht[index].head;
  ht[index].head=temp;
  ht[index].count++;
}
```



Hash Table	
Count	**address**
0	NULL
0	NULL
0	NULL
0	NULL
1	

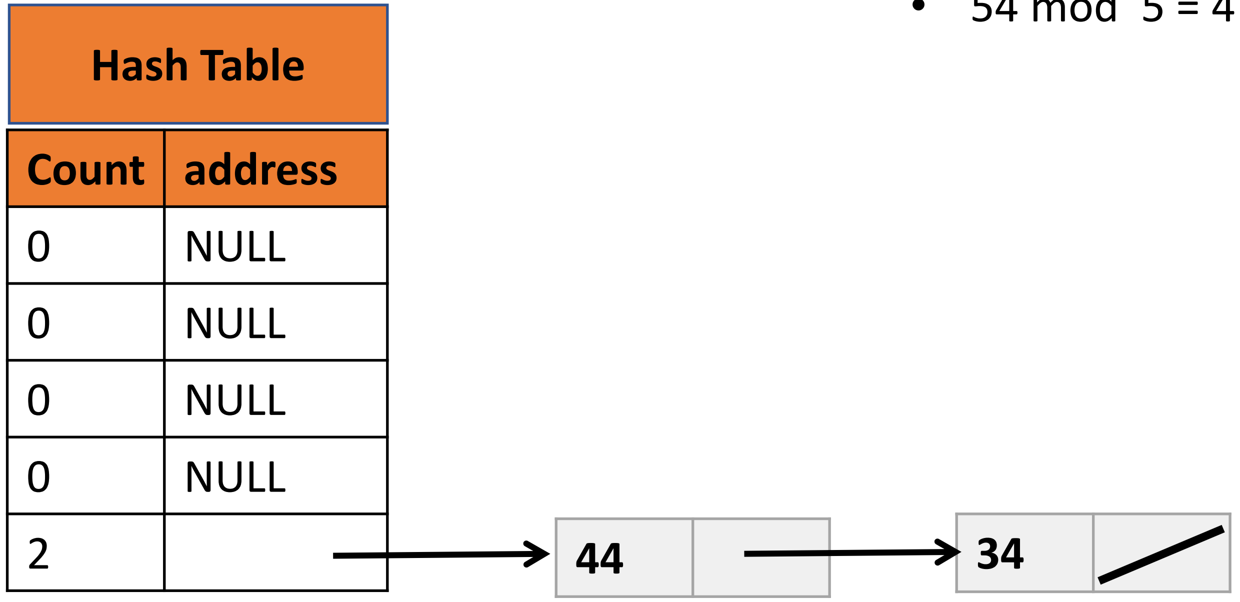
- **34 mod 5 = 4,   34 is stored at index 4.**
- 44 mod 5 = 4,   44 is stored at index 4.
- 54 mod 5 = 4 ,   54 is stored at index 4.

Hashing – Open Addressing / Separate Chaining

// Insert node at the beginning of Singly linked list as shown in the figure.

```
 index=key%size;
  temp->next=ht[index].head;
  ht[index].head=temp;
  ht[index].count++;
}
```

- 34 mod 5 = 4,   34 is stored at index 4.
- **44 mod 5 = 4,   44 is stored at index 4**.
- 54 mod 5 = 4 ,   54 is stored at index 4.



Hash Table

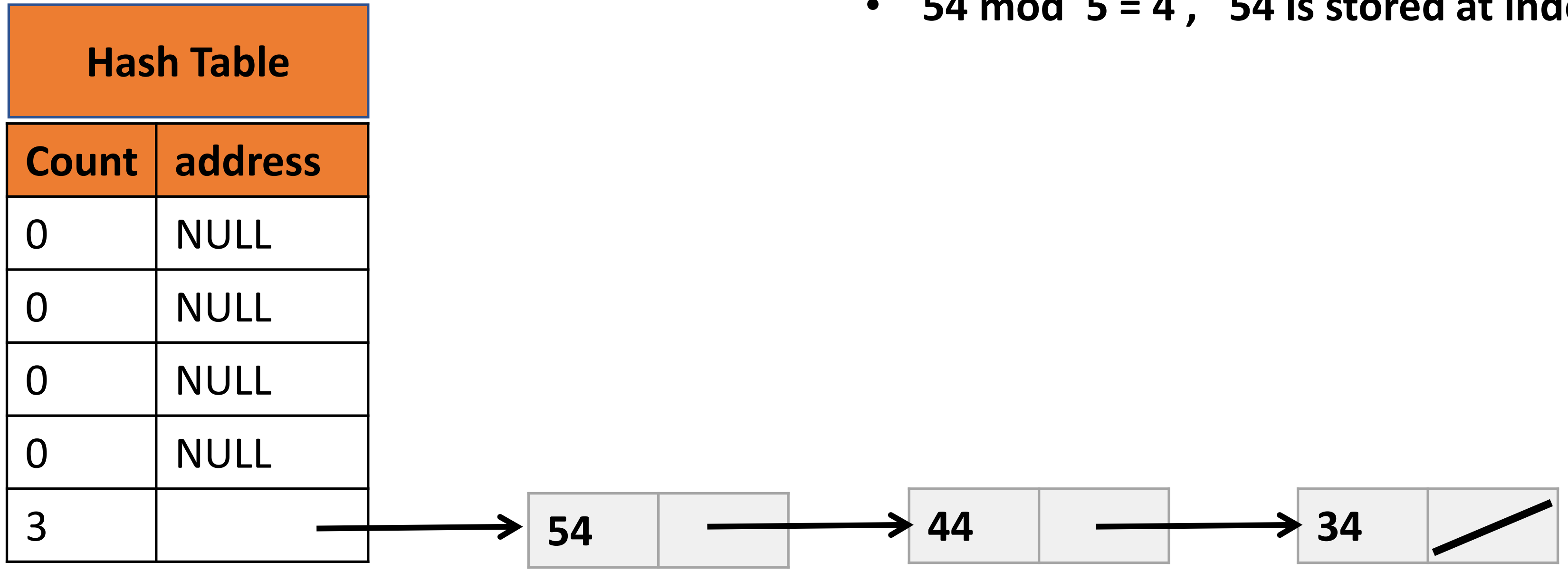
Count	**address**
0	NULL
0	NULL
0	NULL
0	NULL
2	

Hashing – Open Addressing / Separate Chaining

// Insert node at the beginning of Singly linked list as shown in the figure.

```
 index=key%size;
  temp->next=ht[index].head;
  ht[index].head=temp;
  ht[index].count++;
}
```

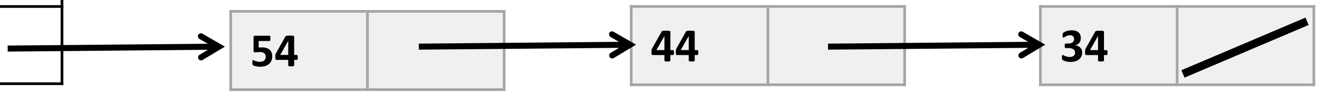
- 34 mod 5 = 4, 34 is stored at index 4.
- 44 mod 5 = 4, 44 is stored at index 4.
- **54 mod 5 = 4 , 54 is stored at index 4.**



Hashing – Open Addressing / Separate Chaining – Display Operation

```
void display(struct hash* ht, int size)
 {
  int i;
  struct node *temp;
  printf("\n");
 for(i=0;i<size;i++)
  {
     printf("%d : ",i)
     if(ht[i].head != NULL)
       {
         temp=ht[i].head;
         while(temp!=NULL)
            {
              printf("%d",temp->key);
              printf("%s->",temp->name);
              temp=temp->next;
            }
       }
      printf("\n");
   }
 }
```

Count	address
0	NULL
0	NULL
0	NULL
0	NULL
3	



Display Output :

```
0   :
1   :
2   :
3   :
4   :   54 ->  44->  34
```

PES
UNIVERSITY
ONLINE

THANK YOU

V R BADRI PRASAD
Department of Computer Science & Engineering

badriprasad@pes.edu