## Program

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

struct Node {
    int data;
    struct Node *next;
};

struct HashTable {
    struct Node *arrayChaining[SIZE];
    int arrayLinearProbing[SIZE];
};

struct Node *createNode(int data) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

struct HashTable *createHashTable() {
    struct HashTable *hashTable = (struct HashTable *)malloc(sizeof(struct HashTable));
    for (int i = 0; i < SIZE; i++) {
        hashTable->arrayChaining[i] = NULL;
        hashTable->arrayLinearProbing[i] = -1;
    }
    return hashTable;
}

void insertChaining(struct HashTable *hashTable, int key) {
    int index = key % SIZE;
    struct Node *newNode = createNode(key);
    if (hashTable->arrayChaining[index] == NULL)
        hashTable->arrayChaining[index] = newNode;
    else {
        struct Node *current = hashTable->arrayChaining[index];
        while (current->next != NULL)
            current = current->next;
        current->next = newNode;
    }
}

void insertLinearProbing(struct HashTable *hashTable, int key) {
    int index = key % SIZE;
    while (hashTable->arrayLinearProbing[index] != -1)
        index = (index + 1) % SIZE;
    hashTable->arrayLinearProbing[index] = key;
}

void displayChaining(struct HashTable *hashTable) {
    for (int i = 0; i < SIZE; i++) {
        printf("Index %d (Chaining): ", i);
        struct Node *current = hashTable->arrayChaining[i];
        while (current != NULL) {
            printf("%d -> ", current->data);
            current = current->next;
        }
        printf("NULL\n");
    }
}
```

Experiment No: 29
Date:

# HASHING

## Aim:

To create a hash table using chaining and linear probing to store and retrieve keys

## Algorithm:

1. Start
2. Define constants:
   SIZE = 10
3. create a structure Node(int data, struct Node *next)
4. create a structure HashTable(struct Node *arrayChaining[SIZE], int arrayLinearProbing[SIZE])
5. Function createNode(data):

```
Allocate memory for newNode
Set newNode->data = data
Set newNode->next = NULL
Return newNode
```

6. Function createHashTable():

```
Allocate memory for hashTable
For each index in array:
    Initialize hashTable->arrayChaining[i]=NULL
    Set hashTable->arrayLinearProbing[i]=-1
Return hashTable
```

7. Function insertChaining(hashTable, key):

```
Compute index = key % SIZE
Create newNode with the key
If hashTable->arrayChaining[index]=NULL:
    hashTable->arrayChaining[index]=newNode
Else:
    Traverse linked list at hashTable->arrayChaining[index] until the end
    Add newNode at the end
End if
```

8. Function insertLinearProbing(hashTable, key):

```c
void displayLinearProbing(struct HashTable *hashTable) {
    for (int i = 0; i < SIZE; i++) {
        printf("Index %d (Linear Probing): %d\n", i, hashTable->arrayLinearProbing[i]);
    }
}

int main() {
    struct HashTable *hashTable = createHashTable();
    int choice, key;
    do {
        printf("\nMenu:\n");
        printf("1. Insert into Chaining Method\n");
        printf("2. Insert into Linear Probing Method\n");
        printf("3. Display Chaining Method\n");
        printf("4. Display Linear Probing Method\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter key to insert using Chaining Method: ");
                scanf("%d", &key);
                insertChaining(hashTable, key);
                break;
            case 2:
                printf("Enter key to insert using Linear Probing Method: ");
                scanf("%d", &key);
                insertLinearProbing(hashTable, key);
                break;
            case 3:
                printf("Hash Table using Chaining Method:\n");
                displayChaining(hashTable);
                break;
            case 4:
                printf("Hash Table using Linear Probing Method:\n");
                displayLinearProbing(hashTable);
                break;
            case 5:
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }
    } while (choice != 5);
    return 0;
}
```

## Output

```
Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 1
Enter key to insert using Chaining Method: 3

Menu:
```

```
Compute index = key % SIZE
While hashTable->arrayLinearProbing[index] is not -1:
    Increment index circularly using (index + 1) % SIZE
Assign key to hashTable->arrayLinearProbing[index]
End while
```

9. Function displayChaining(hashTable):

```
For each index in hashTable->arrayChaining:
    Print "Index <index> (Chaining): "
    Traverse the linked list at that index
    Print data of each node followed by " -> "
    Print "NULL" at the end
End for
```

10. Function displayLinearProbing(hashTable):

```
For each index in hashTable->arrayLinearProbing:
    Print "Index <index> (Linear Probing): " and value at that index
End for
```

11. Main function:

```
Create a hashTable
Do:
    Display menu with options:
        1. Insert into Chaining Method
        2. Insert into Linear Probing Method
        3. Display Chaining Method
        4. Display Linear Probing Method
        5. Exit
    Read choice from user
    Switch choice:
        Case 1:
            Prompt user for key to insert using Chaining Method
            Call insertChaining(hashTable, key)
        Case 2:
            Prompt user for key to insert using Linear Probing Method
            Call insertLinearProbing(hashTable, key)
        Case 3:
            Print "Hash Table using Chaining Method"
            Call displayChaining(hashTable)
        Case 4:
            Print "Hash Table using Linear Probing Method"
            Call displayLinearProbing(hashTable)
        Case 5:
            Print "Exiting the program."
        Default:
            Print "Invalid choice. Please enter a valid option."
While choice is not 5
```

12. Stop

```
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 1
Enter key to insert using Chaining Method: 5

Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 3
Hash Table using Chaining Method:
Index 0 (Chaining): NULL
Index 1 (Chaining): NULL
Index 2 (Chaining): NULL
Index 3 (Chaining): 3 -> NULL
Index 4 (Chaining): NULL
Index 5 (Chaining): 5 -> NULL
Index 6 (Chaining): NULL
Index 7 (Chaining): NULL
Index 8 (Chaining): NULL
Index 9 (Chaining): NULL

Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 2
Enter key to insert using Linear Probing Method: 2

Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 2
Enter key to insert using Linear Probing Method: 4

Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 4
Hash Table using Linear Probing Method:
Index 0 (Linear Probing): -1
Index 1 (Linear Probing): -1
Index 2 (Linear Probing): 2
Index 3 (Linear Probing): -1
Index 4 (Linear Probing): 4
Index 5 (Linear Probing): -1
Index 6 (Linear Probing): -1
Index 7 (Linear Probing): -1
Index 8 (Linear Probing): -1
Index 9 (Linear Probing): -1
```

```
Menu:
1. Insert into Chaining Method
2. Insert into Linear Probing Method
3. Display Chaining Method
4. Display Linear Probing Method
5. Exit
Enter your choice: 5
Exiting the program.
```

## Result:

Program has been executed successfully and obtained the output.