Experiment 26
Hashing with Chaining

Date: 12-02-2021

Aim: Implementation of Hash table with chaining as the collition

Data Structure Used: Hash table

Algorithm for Inserting an element (insert)

Input: Element to be inserted, v. The hash table to which the element is inserted HT, and the hash function h(v)
Output: The element inserted in the hash table
Data Structure: Hash table

Steps:

```
Step 1: Start
Step 2: key = h(v)
Step 3: new = GetNode(Node)
Step 4: new→data = v
Step 5: new →next =NULL
Step 6: if(HT[key]!=NULL) then
        Step 4: HT[key] = new
Step 7: else
        Step 1: ptr = HT[key]
        Step 2: while(ptr→next != NULL) do
                Step 1: ptr = ptr→ next
        Step 3: EndWhile
        Step 4: ptr →next = new
Step 8: endif
Step 9: Stop
```

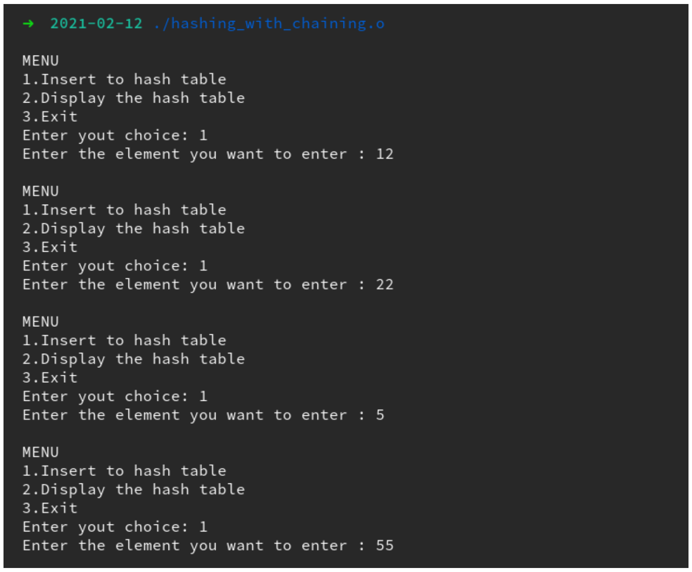
Program Code:

```
/***********************************
 * Implementiong of Hashing with
 * chaining
 * Done By: Rohit Karunakaran
 * ******************************/

#include<stdio.h>
#include<stdlib.h>

#define SIZE 10

typedef struct hash_node{
    int data;
    struct hash_node *next;
} node;

void insert(node ***ht, int e){
    int pos = e%SIZE;
    node **hash_table = *ht;
    node *new_node = (node*)malloc(sizeof(node));
    if(new_node !=NULL){
        new_node->data = e;
        new_node->next=NULL;
        if(hash_table[pos]==NULL){
            hash_table[pos] = new_node;
        }
        else{
            node *ptr = hash_table[pos];
            while(ptr->next!=NULL)ptr = ptr->next;
            ptr->next=new_node;
        }
    }
}

void show_the_hash_table(node **ht){
    int i;
    for(i=0;i<SIZE;i++){
        node *ptr = ht[i];
        if(ptr!=NULL){
            while(ptr->next!=NULL){
                printf("%d -> ",ptr->data);
                ptr = ptr->next;
            }
            printf("%d\n",ptr->data);
        }
    }
}

int main(){
    node **hash_table=(node**)calloc(sizeof(node),SIZE);
    int RUN=1;
    int elem;
    int c;

    while(RUN){
        printf("\nMENU\n");
        printf("1.Insert to hash table\n");
        printf("2.Display the hash table\n");
        printf("3.Exit\n");
        printf("Enter yout choice: ");

        scanf("%d",&c);

        switch(c){

            case 1:
                printf("Enter the element you want to enter : ");
                scanf("%d",&elem);
                insert(&hash_table, elem);
                break;
            case 2:
                printf("The hash table is : \n");
                show_the_hash_table(hash_table);
                break;
            case 3:
                RUN=0;
                break;
        }
    }
    return 0;
}
```

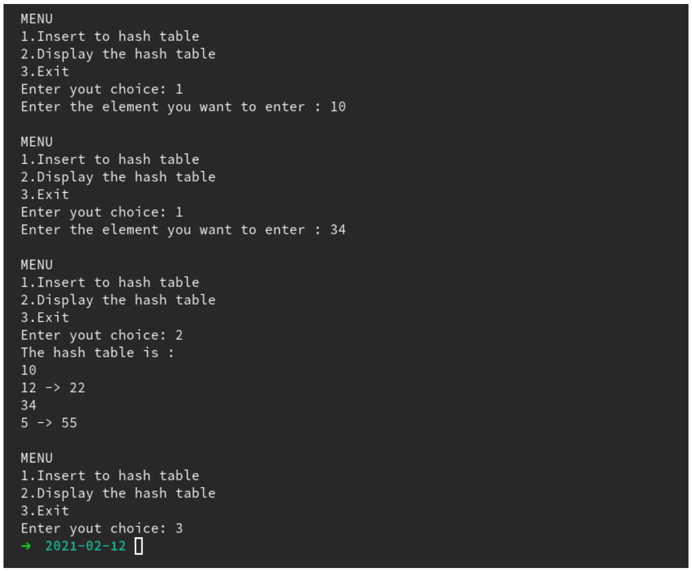
Result:

The program was successfully compiled and the required output was obtained.

<u>Sample input output:</u>



```
➜  2021-02-12 ./hashing_with_chaining.o

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 12

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 22

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 5

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 55
```



```
MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 10

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 34

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 2
The hash table is :
10
12 -> 22
34
5 -> 55

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 3
➜  2021-02-12
```

Experiment 27
Hashing with Linear Probing

Date: 12-02-2021

Aim: Implementation of Hash table with linear probing as the collition resolution method

Data Structure Used: Hash table

Algorithm for Inserting an element (insert)

Input: Element to be inserted, v. The hash table to which the element is inserted HT with the hash function h(v)
Output: The element inserted in the hash table
Data Structure: Hash table

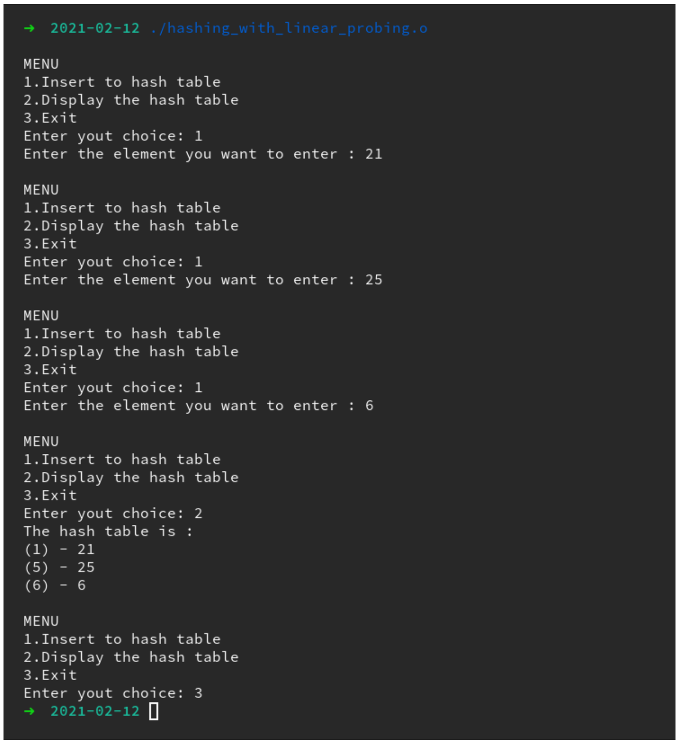
Steps:

```
Step 1: Start
Step 2: key = h(v)
Step 3: new = GetNode(Node)
Step 4: new→value = v
Step 6: if(HT[key]!=NULL) then
        Step 4: HT[key] = new
Step 7: else
        Step 1: new_key = key+1
        Step 2: while(new_key!=key and HT[new_key]!=NULL) do
                Step 1: new_key = new_key+1
        Step 3: EndWhile
        Step 4: if(new_key == key) then
                Step 1: Print “Insertion not possible”
                Step 2: Exit
        Step 5: else
                HT[new_key] = new
        Step 6: endif
Step 8: endif
Step 9: Stop
```

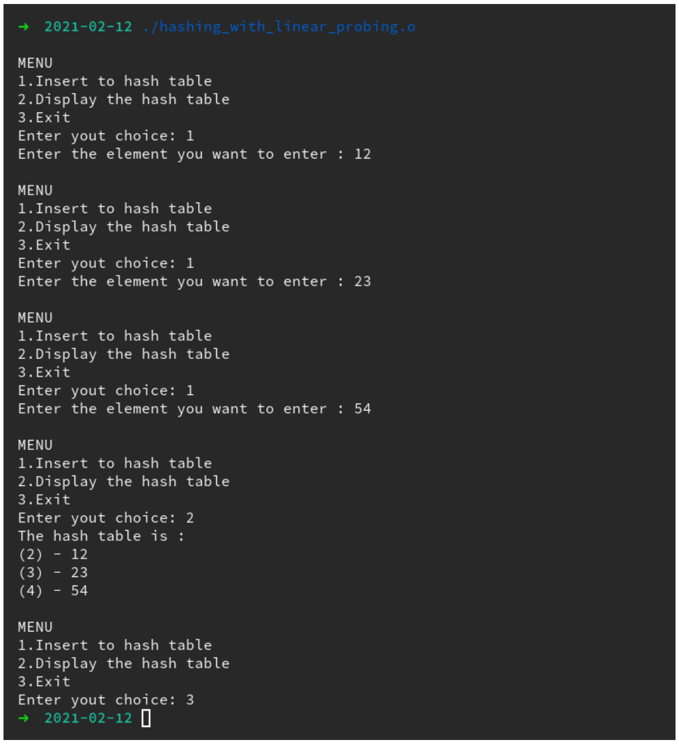
Program Code:

```
/************************************
 * Hashing with linear probing as the
 * collision resolution method
 *
 * Done By: Rohit Karunakaran
 ***********************************/


#include<stdio.h>
#include<stdlib.h>

#define SIZE 10

void insert(int ***hash_table, int e,int i){
    if(i<=SIZE){
        int pos = (e%SIZE+i)%SIZE;
        int **ht = *hash_table;

        if(ht[pos]==NULL){
            int* node = (int*) malloc(sizeof(int));
            *node = e;
            ht[pos] = node;
        }
        else{
            insert(hash_table,e,i+1);
        }
    }
    else{
        printf("INSERTION NOT POSSIBLE!!!!!!\nThe Hash Table is full\n");
        return;
    }

}

void show_the_hash_table(int **ht){
    int i;
    for(i=0;i<SIZE;i++){
        int *ptr = ht[i];
        if(ptr!=NULL){
            printf("(%d) - %d\n",i,*ptr);
        }
    }
}

int main(){
    int **hash_table=(int**)calloc(sizeof(int*),SIZE);
    int RUN=1;
    int elem;
    int c;

    while(RUN){
        printf("\nMENU\n");
        printf("1.Insert to hash table\n");
        printf("2.Display the hash table\n");
        printf("3.Exit\n");
        printf("Enter yout choice: ");

        scanf("%d",&c);

        switch(c){

            case 1:
                printf("Enter the element you want to enter : ");
                scanf("%d",&elem);
                insert(&hash_table, elem,0);
                break;
            case 2:
                printf("The hash table is : \n");
                show_the_hash_table(hash_table);
                break;
            case 3:
                RUN=0;
                break;
        }

    }

    for(int i = 0; i<SIZE;i++){
        free(*(hash_table+i));
    }
    free(hash_table);

    return 0;
}
```

Sample input output:



```
➜  2021-02-12 ./hashing_with_linear_probing.o

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 21

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 25

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 6

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 2
The hash table is :
(1) - 21
(5) - 25
(6) - 6

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 3
➜  2021-02-12
```



```
➜  2021-02-12 ./hashing_with_linear_probing.o

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 12

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 23

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 1
Enter the element you want to enter : 54

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 2
The hash table is :
(2) - 12
(3) - 23
(4) - 54

MENU
1.Insert to hash table
2.Display the hash table
3.Exit
Enter yout choice: 3
➜  2021-02-12
```

Result: The program is successfully compiled and the required output is obtained