ACADEMY OF TECHNOLOGY

NAME : PRIYADARSHAN GHOSH

COLLEGE ROLL NO: 72

UNIVERSITY ROLL NO: 16900319072

DEPARTMENT: ECE-1(Y)

SEMESTER: 3rd

PAPER CODE : ES-CS391

➢ Laboratory *Assignment #17*

A.TO IMPLEMENT LINEAR PROBING METHOD IN COLLISION RESOLUTION TECHNIQUE.

Ans:

```
#include <stdio.h>
#include <conio.h>

int hasht(int key, int tsize)
{
        int i ;
        i = key%tsize ;
        return i;
}

int linearProbe(int key, int tsize)
{
        int i ;
        i = (key+1)%tsize ;
        return i ;
}

int main()
{
    int key, arr[20], hash[20], tsize, i, n, k;

    printf ("\nEnter the size of the hash table:  ");
        scanf ("%d",&tsize);
```

```
    printf ("\nEnter the number of elements: ");
        scanf ("%d",&n);

    for (i=0;i<tsize;i++)
                hash[i]=-1 ;

    printf ("Enter Elements: ");
    for (i=0;i<n;i++)
                scanf("%d",&arr[i]);

        for (i=0;i<tsize;i++)
        hash[i]=-1 ;
    for(k=0; k<n; k++)
        {
                key=arr[k] ;
                i = hasht(key, tsize);
                while (hash[i]!=-1)
                i = linearProbe(i, tsize);
                hash[i]=key ;
    }
    printf("\nThe elements in the array are: ");
    for (i=0;i<tsize;i++)
                printf("\n Element at position %d: %d",i,hash[i]);
}
```

OUTPUT =>

```
Enter the size of the hash table:  8
```

```
Enter the number of elements: 1
Enter Elements: 9

The elements in the array are:
        Element at position 0: -1
        Element at position 1: 9
        Element at position 2: -1
        Element at position 3: -1
        Element at position 4: -1
        Element at position 5: -1
        Element at position 6: -1
        Element at position 7: -1
--------------------------------
Process exited after 7.832 seconds with return value 0
Press any key to continue . . .
```

B. TO IMPLEMENT QUADRATIC PROBING METHOD IN COLLISION RESOLUTION TECHNIQUE.

Ans:

```
#include <stdio.h>
#include <conio.h>

int hasht(int key, int tsize)
{
        int i ;
        i = key%tsize ;
        return i;
```

```
}

int quadraticProbe(int key, int j, int tsize)
{
        int i ;
        i = (key+(j*j))%tsize ;
        return i ;
}

int main()
{
    int key, arr[20], hash[20], tsize, i, n, j, k;

    printf ("\nEnter the size of the hash table:  ");
        scanf ("%d",&tsize);

    printf ("\nEnter the number of elements: ");
        scanf ("%d",&n);

    for (i=0; i<tsize; i++)
                hash[i]=-1 ;

    printf ("Enter Elements: ");
    for (i=0; i<n; i++)
                scanf("%d",&arr[i]);

    for (i=0; i<tsize; i++)
```

```
        hash[i]=-1 ;
    for(k=0;k<n;k++)
    {
        j=1;
        key=arr[k];
        i = hasht(key, tsize);
        while (hash[i]!=-1)
        {
                i=hasht(key, tsize);
                i = quadraticProbe(i,j, tsize);
                j++ ;
        }
        hash[i]=key ;
        }
    printf("\nThe elements in the array are: ");
    for (i=0;i<tsize;i++)
                printf("\n Element at position %d: %d",i,hash[i]);
}
```

OUTPUT =>

```
Enter the size of the hash table:  9

Enter the number of elements: 1
Enter Elements: 7

The elements in the array are:
        Element at position 0: -1
        Element at position 1: -1
```

```
        Element at position 2: -1
        Element at position 3: -1
        Element at position 4: -1
        Element at position 5: -1
        Element at position 6: -1
        Element at position 7: 7
        Element at position 8: -1
--------------------------------
Process exited after 8.639 seconds with return value 0
Press any key to continue . . .
```

C. TO IMPLEMENT DOUBLE HASHING METHOD IN COLLISION RESOLUTION TECHNIQUE.

Ans:

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 10
#define PRIME 7

int h[TABLE_SIZE]={NULL};

void insert()
{

 int key,index,i,flag=0,h1key,h2key;
 printf("\nenter a value to insert into hash table\n");
 scanf("%d",&key);
```

```
 h1key=key%TABLE_SIZE;
 h2key=PRIME-(key%PRIME);
 for(i=0;i<TABLE_SIZE;i++)
    {

     index=(h1key+i*h2key)%TABLE_SIZE;

     if(h[index] == NULL)
     {
        h[index]=key;
         break;
     }

    }

    if(i == TABLE_SIZE)

     printf("\nelement cannot be inserted\n");
}
void search()
{

 int key,index,i,flag=0,hkey;
 printf("\nenter search element\n");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE ; i++)
```

```
 {
    index=(hkey+i)%TABLE_SIZE;
    if(h[index]==key)
    {
      printf("value is found at index %d",index);
      break;
    }
  }
  if(i == TABLE_SIZE)
    printf("\n value is not found\n");
}
void display()
{

  int i;

  printf("\nelements in the hash table are \n");

  for(i=0;i< TABLE_SIZE; i++)

  printf("\nat index %d \t value =  %d",i,h[i]);

}
 int main()
{
    int opt,i;
    while(1)
```

```
    {
        printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");
        scanf("%d",&opt);
        switch(opt)
        {
            case 1:
                insert();
                break;
            case 2:
                display();
                break;
            case 3:
                search();
                break;
            case 4:exit(0);
        }
    }
    return 0;
}
```

OUTPUT =>

```
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
4

Press 1. Insert  2. Display     3. Search       4.Exit
```

1

enter a value to insert into hash table
6

Press 1. Insert 2. Display 3. Search 4.Exit
1

enter a value to insert into hash table
7

Press 1. Insert 2. Display 3. Search 4.Exit
1

enter a value to insert into hash table
9

Press 1. Insert 2. Display 3. Search 4.Exit
1

enter a value to insert into hash table
9

element cannot be inserted

Press 1. Insert 2. Display 3. Search 4.Exit
1

```
enter a value to insert into hash table
4

Press 1. Insert  2. Display     3. Search       4.Exit
2

elements in the hash table are

at index 0       value =  4
at index 1       value =  0
at index 2       value =  0
at index 3       value =  0
at index 4       value =  4
at index 5       value =  0
at index 6       value =  6
at index 7       value =  7
at index 8       value =  0
at index 9       value =  9
Press 1. Insert  2. Display     3. Search       4.Exit
3

enter search element
7
value is found at index 7
Press 1. Insert  2. Display     3. Search       4.Exit
1
```

```
enter a value to insert into hash table
45

Press 1. Insert  2. Display     3. Search       4.Exit
2

elements in the hash table are

at index 0       value =  4
at index 1       value =  0
at index 2       value =  0
at index 3       value =  0
at index 4       value =  4
at index 5       value =  45
at index 6       value =  6
at index 7       value =  7
at index 8       value =  0
at index 9       value =  9
Press 1. Insert  2. Display     3. Search       4.Exit
3

enter search element
9
value is found at index 9
Press 1. Insert  2. Display     3. Search       4.Exit
4
```

```
--------------------------------
Process exited after 87.88 seconds with return value 0
Press any key to continue . . .
```

D. TO IMPLEMENT SEPARATE CHAINING METHOD IN COLLISION RESOLUTION TECHNIQUE.

Ans:

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

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

struct node *table[SIZE];

void insert()
{

      int value;
      printf("\nenter a value to insert into hash table\n");
      scanf("%d",&value);
```

```
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;


    int key = value % SIZE;


    if(table[key] == NULL)
        table[key] = newNode;

    else
    {
        struct node *temp = table[key];
        while(temp->next)
        {
            temp = temp->next;
        }

        temp->next = newNode;
    }
}

void display()
{
```

```
    int i;

    for(i = 0; i < SIZE; i++)
    {
        struct node *temp = table[i];
        printf("table[%d]-->",i);
        while(temp)
        {
            printf("%d -->",temp->data);
            temp = temp->next;
        }
        printf("NULL\n");
    }
}
void search()
{
        int value,i,key;
        printf("\nenter search element\n");
        scanf("%d",&value);
        key=value % SIZE;
        struct node *temp = table[key];

        while(temp!=NULL)
        {
          if(temp->data==value)
```

```
        {
                printf("ELEMENT FOUND IN LOCATION %d",key);
                return;
            }
         temp = temp->next;
      }
      printf("ELEMENT NOT FOUND");
      return;
   }



int main()
{
   int opt,i;

   for(i = 0; i < SIZE; i++)
      table[i] = NULL;

   while(1)
   {
      printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");
      scanf("%d",&opt);
      switch(opt)
      {
```

```
            case 1:
                insert();
                break;
            case 2:
                display();
                break;
            case 3:
                search();
                break;
            case 4:exit(0);
        }
    }
    return 0;
}
```

OUTPUT =>

```
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
2

Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
5
```

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

7

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

9

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

6

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table

87

Press 1. Insert 2. Display 3. Search 4.Exit

1

enter a value to insert into hash table
45

Press 1. Insert 2. Display 3. Search 4.Exit
1

enter a value to insert into hash table
24

Press 1. Insert 2. Display 3. Search 4.Exit
2
table[0]-->NULL
table[1]-->NULL
table[2]-->2 -->NULL
table[3]-->NULL
table[4]-->24 -->NULL
table[5]-->5 -->45 -->NULL
table[6]-->6 -->NULL
table[7]-->7 -->87 -->NULL
table[8]-->NULL
table[9]-->9 -->NULL

Press 1. Insert 2. Display 3. Search 4.Exit
1

enter a value to insert into hash table

```
23

Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
67

Press 1. Insert  2. Display     3. Search       4.Exit
2
table[0]-->NULL
table[1]-->NULL
table[2]-->2 -->NULL
table[3]-->23 -->NULL
table[4]-->24 -->NULL
table[5]-->5 -->45 -->NULL
table[6]-->6 -->NULL
table[7]-->7 -->87 -->67 -->NULL
table[8]-->NULL
table[9]-->9 -->NULL

Press 1. Insert  2. Display     3. Search       4.Exit
3

enter search element
24
ELEMENT FOUND IN LOCATION 4
```

```
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
43

Press 1. Insert  2. Display     3. Search       4.Exit
2
table[0]-->NULL
table[1]-->NULL
table[2]-->2 -->NULL
table[3]-->23 -->43 -->NULL
table[4]-->24 -->NULL
table[5]-->5 -->45 -->NULL
table[6]-->6 -->NULL
table[7]-->7 -->87 -->67 -->NULL
table[8]-->NULL
table[9]-->9 -->NULL

Press 1. Insert  2. Display     3. Search       4.Exit
3

enter search element
43
ELEMENT FOUND IN LOCATION 3
Press 1. Insert  2. Display     3. Search       4.Exit
4
```

```
--------------------------------

Process exited after 68.07 seconds with return value 0

Press any key to continue . . .
```