

Practical 9

AIM:

HASH TABLE IMPLEMENTATION

Implement a hash table data structure using different hash function and collision resolution techniques such as chaining and open addressing.

Program:-

```
#include <stdio.h>

// Creating array Globaly
int Queue[5];
int front = -1, rear = -1, data;

// FUNCTION FOR ENQUEUE
int enqueue()
{
    if((rear + 1) % 5 == front){
        printf("The Queue is Overflow.\n");
    }else if(front == -1 && rear == -1){
        front = 0;
        rear = 0;
        printf("Enter the data.\n");
        scanf("%d", &data);
        Queue[rear] = data;
    }else{
        printf("Enter the data.\n");
        scanf("%d", &data);
        rear = (rear + 1) % 5;
        Queue[rear] = data;
    }
    return 0;
}

// FUNCTION FOR DEQUEUE
int dequeue()
```

```

{

    if(front == -1 && rear == -1 ){

        printf("The Queue is Underflow.\n");

    }else if(front == rear){

        printf("The Queue is Underflow.\n");

        front = rear = -1;

    }else{

        printf("The deleting element is %d.\n", Queue[front]);

        front = (front + 1) % 5;

    }

    return 0;

}

void display()

{

    if (front == -1)

    {

        // Checking the queue is empty or not.

        printf("The Queue is empty so, can not print the element.\n");

    }

    else

    {

        // printing the elements in the Queue

        int i = front;

        while (1)

        {

            printf("%d\t", Queue[i]);

            if (i == rear)

                break;    // Stop when we reach the rear

            i = (i + 1) % 5; // Move to the next index in circular manner

        }

        printf("\n");

    }

}

```

```
// MAIN FUNCTION

int main()

{

    int choice;

    printf("Queue Implementation\n");

    printf("Choices\n1.Enqueue\t2.Dequeue\t3.Print\t4.Exit\n");

    do

    {

        printf("Enter a valid choice\n");

        scanf("%d", &choice);


        switch (choice)

        {

            case 1:

                enqueue();

                break;

            case 2:

                dequeue();

                break;

            case 3:

                display();

                break;

            case 4:

                printf("You exited the Program successfully.");

                break;

            default:

                printf("Please enter a valid choice as mention!\n");

                break;

        }

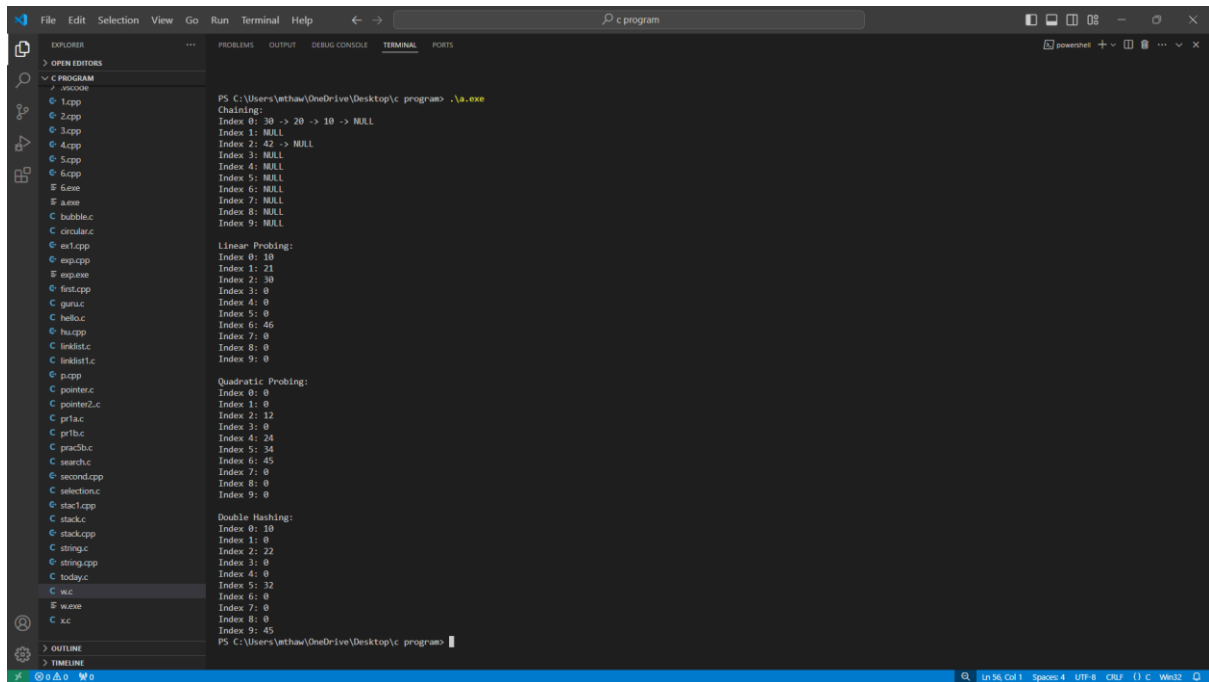
    } while (choice != 4);

}
```

```
return 0;

}
```

Output:-



```
PS C:\Users\ethaw\OneDrive\Desktop\c program> .\a.exe
Chaining:
Index 0: 30 -> 20 -> 10 -> NULL
Index 1: NULL
Index 2: 42 -> NULL
Index 3: NULL
Index 4: NULL
Index 5: NULL
Index 6: NULL
Index 7: NULL
Index 8: NULL
Index 9: NULL

Linear Probing:
Index 0: 10
Index 1: 21
Index 2: 30
Index 3: 0
Index 4: 0
Index 5: 0
Index 6: 46
Index 7: 0
Index 8: 0
Index 9: 0

Quadratic Probing:
Index 0: 0
Index 1: 0
Index 2: 12
Index 3: 0
Index 4: 24
Index 5: 34
Index 6: 45
Index 7: 0
Index 8: 0
Index 9: 0

Double Hashing:
Index 0: 10
Index 1: 0
Index 2: 22
Index 3: 0
Index 4: 0
Index 5: 32
Index 6: 0
Index 7: 0
Index 8: 0
Index 9: 45

PS C:\Users\ethaw\OneDrive\Desktop\c program>
```

Github link :- <https://github.com/MayurThaware122/DSA>