**Hashing with Linear Probing Technique**

#include <stdio.h>  
  #include <stdlib.h>  
  #include <string.h>  
  
  int tableSize = 0, totEle = 0;  
  struct node \*hashTable = NULL;  
  
  struct node {  
        int age, key;  
        char name[100];  
        int marker;  
  };  
  
  void insertInHash(int key, char \*name, int age) {  
        int hashIndex = key % tableSize;  
        if (tableSize == totEle) {  
                printf("Can't perform Insertion..Hash Table is full!!");  
                return;  
        }  
        while (hashTable[hashIndex].marker == 1) {  
                hashIndex = (hashIndex + 1)%tableSize;  
        }  
        hashTable[hashIndex].key = key;  
        hashTable[hashIndex].age = age;  
        strcpy(hashTable[hashIndex].name, name);  
        hashTable[hashIndex].marker = 1;  
        totEle++;  
        return;  
  }

  void deleteFromHash(int key) {

        int hashIndex = key % tableSize, count = 0, flag = 0;

        if (totEle == 0) {

                printf("Hash Table is Empty!!\n");

                return;

        }

        while (hashTable[hashIndex].marker != 0 && count <= tableSize) {

                if (hashTable[hashIndex].key == key) {

                        hashTable[hashIndex].key = 0;

                        /\* set marker to -1 during deletion operation\*/

                        hashTable[hashIndex].marker = -1;

                        hashTable[hashIndex].age = 0;

                        strcpy(hashTable[hashIndex].name, "\0");

                        totEle--;

                        flag = 1;

                        break;

                }

                hashIndex = (hashIndex + 1)%tableSize;

                count++;

        }

        if (flag)

                printf("Given data deleted from Hash Table\n");

        else

                printf("Given data is not available in Hash Table\n");

        return;

  }

  void searchElement(int key) {

        int hashIndex = key % tableSize, flag = 0, count = 0;

        if (totEle == 0) {

                printf("Hash Table is Empty!!");

                return;

        }

        while (hashTable[hashIndex].marker != 0 && count <= tableSize) {

                if (hashTable[hashIndex].key == key) {

                        printf("Voter ID : %d\n", hashTable[hashIndex].key);

                        printf("Name     : %s\n", hashTable[hashIndex].name);

                        printf("Age      : %d\n", hashTable[hashIndex].age);

                        flag = 1;

                        break;

                }

                hashIndex = (hashIndex + 1)%tableSize;

        }

        if (!flag)

                printf("Given data is not present in hash table\n");

        return;

  }

  void display() {

        int i;

        if (totEle == 0) {

                printf("Hash Table is Empty!!\n");

                return;

        }

        printf("Voter ID     Name           Age    Index \n");

        printf("-----------------------------------------\n");

        for (i = 0; i < tableSize; i++) {

                if (hashTable[i].marker == 1) {

                        printf("%-13d", hashTable[i].key);

                        printf("%-15s", hashTable[i].name);

                        printf("%-7d", hashTable[i].age);

                        printf("%d\n", i);

                }

        }

        printf("\n");

        return;

  }

  int main() {

        int key, age, ch;

        char name[100];

        printf("Enter the no of elements:");

        scanf("%d", &tableSize);

        hashTable = (struct node \*)calloc(tableSize, sizeof(struct node));

        while (1) {

                printf("1. Insertion\t2. Deletion\n");

                printf("3. Searching\t4. Display\n");

                printf("5. Exit\nEnter ur choice:");

                scanf("%d", &ch);

                switch (ch) {

                        case 1:

                                printf("Enter the key value:");

                                scanf("%d", &key);

                                getchar();

                                printf("Name:");

                                fgets(name, 100, stdin);

                                name[strlen(name) - 1] = '\0';

                                printf("Age:");

                                scanf("%d", &age);

                                insertInHash(key, name, age);

                                break;

                        case 2:

                                printf("Enter the key value:");

                                scanf("%d", &key);

                                deleteFromHash(key);

                                break;

                        case 3:

                                printf("Enter the key value:");

                                scanf("%d", &key);

                                searchElement(key);

                                break;

                        case 4:

                                display();

                                break;

                        case 5:

                                exit(0);

                        default:

                                printf("U have entered wrong Option!!\n");

                                break;

                }

        }

        return 0;

  }

**Output (Closed Hashing - Linear probing Example):**

  jp@jp-VirtualBox:~/cpgms/data\_structures$ ./a.out  
  Enter the no of elements:3  
  1. Insertion 2. Deletion  
  3. Searching 4. Display  
  5. Exit  
  Enter ur choice:1  
  Enter the key value:1  
  Name:Harry    
  Age:23  
  1. Insertion 2. Deletion  
  3. Searching 4. Display  
  5. Exit  
  Enter ur choice:1  
  Enter the key value:2  
  Name:Ram  
  Age:24  
  1. Insertion 2. Deletion  
  3. Searching 4. Display  
  5. Exit