ASSIGNMENT NO 5

Implement all the functions of a dictionary (ADT) using hashing and handle collisions using separate chaining using linked list Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable, and Keys must be unique. Standard Operations: Insert(key, value), Find(key), Delete(key)

```
//************SEPARATE CHAINING***************
#include <iostream>
using namespace std;
struct Node
{
    string key;
    string mean;
    struct Node *next;
};

class HashTable
{
public:
    Node *ht[10];
    HashTable()
    {
        for(int i=0;i<10;i++)
        {
            ht[i]=0;
        }
    }
    void insert(string key,string mean);
    void search(string key);
    int isPresent(string key);
    void deleteKey(string key);
    int hashfun(string key);

    void display(){
    for(int i=0;i<10;i++)
    {
        cout<<i<<" : ";
        Node *t=ht[i];
        while(t!=NULL)
        {
            cout<<t->key<<" "<<t->mean;
            t=t->next;
        }
        cout<<endl;
    }
    }
```

```
};
int HashTable::hashfun(string key)
{
    return key[0]%10;
}
int HashTable::isPresent(string key)
{

    for(int i=0;i<10;i++)
    {
        Node *t=ht[i];
        while(t!=NULL)
        {
            if(t->key==key)
            {
                return 0;
            }
            t=t->next;
        }
        return 1;
    }
}
void HashTable::insert(string key,string mean)
{
    Node *p=new Node();
    int x=isPresent(key);
    p->key=key;
    p->mean=mean;
    p->next=NULL;
    if(x==1)
    {
        int i=hashfun(key);
        if(ht[i]==NULL)
        {
            ht[i]=p;
        }
        else
        {
            Node *q=ht[i];
            while(q->next!=NULL)
            {
                q=q->next;
            }
            q->next=p;
        }
```

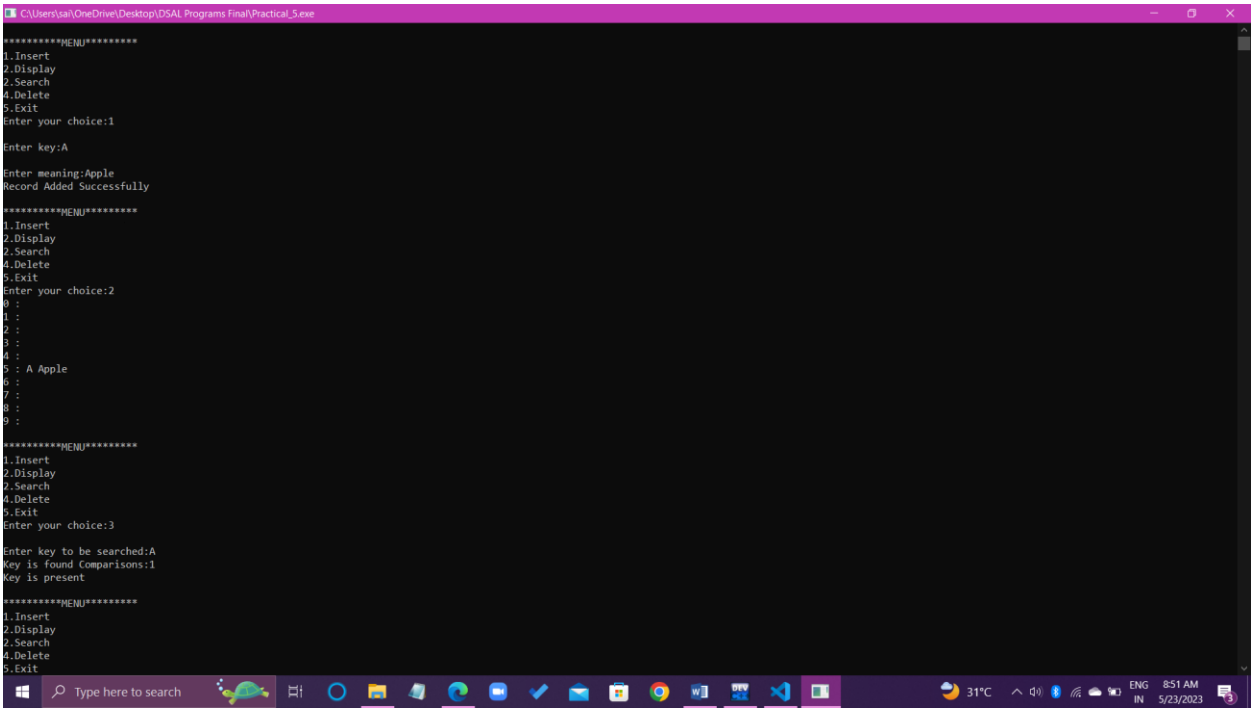
```
        cout<<"Record Added Successfully"<<endl;
    }
    else
    {
        cout<<"Key is already present";
    }
}
void HashTable::search(string key)
{
    int i=hashfun(key);
    int flag=0,count=0;
    Node *t=ht[i];
    if(t==NULL)
    {
        count++;
        cout<<"Key is not present"<<"Comparisons:"<<count<<endl;
    }
    else
    {
        while(t!=NULL)
        {
            count++;
            if(t->key==key)
            {
                cout<<"Key is found"<<" Comparisons:"<<count<<endl;
                cout<<"Key is present"<<endl;
                break;
            }
            t=t->next;
        }
    }
}
void HashTable::deleteKey(string key)
{
    int i=hashfun(key);
    Node *t=ht[i];
    int flag=0;
    if(t==NULL)
    {
        cout<<"Key is not present"<<endl;
    }
    else
    {
        Node *q=ht[i];
```

```
        // If value to be deleted is present in the hashtable and not in the
links next to them
        if(t->key==key)
        {
            Node *p=t->next;
            ht[i]=p;
            delete p;
            cout<<"Key is deleted";
            flag=1;
        }

        //If the value to be deleted is present in the links and not in the
hashtable
        else
        {
            Node *s;
            while(q->next!=NULL)
            {
                s=q;
                q=q->next;
                if(q->key==key)
                {
                    Node *p=q->next;
                    s->next=p;
                    delete q;
                    cout<<"key is deleted"<<endl;
                    flag=1;
                    break;
                }
            }
        }
        if(flag==0)
        {
            cout<<"Key is not present";
        }
    }
}

int main()
{
    int ch;
    string key,mean;
    HashTable t;
    do
    {
```

```
        cout<<"\n**********MENU*********";
        cout<<"\n1.Insert";
        cout<<"\n2.Display";
        cout<<"\n2.Search";
        cout<<"\n4.Delete";
        cout<<"\n5.Exit";
        cout<<"\nEnter your choice:";
        cin>>ch;
        switch(ch)
        {
        case 1:
            cout<<"\nEnter key:";
            cin>>key;
            cout<<"\nEnter meaning:";
            cin.ignore();
            getline(cin,mean);
            t.insert(key,mean);
            break;
        case 2:
            t.display();
            break;
        case 3:
            cout<<"\nEnter key to be searched:";
            cin.ignore();
            getline(cin,key);
            t.search(key);
            break;
        case 4:
            cout<<"\nEnter key to be deleted:";
            cin.ignore();;
            getline(cin,key);
            t.deleteKey(key);
            break;
        case 5:
            cout<<"Thank you for using this program!";
            exit(0);
            break;
        default:
            cout<<"Enter correct choice"<<endl;
            break;
        }
    }while(ch!=4);
    return 0;
}
```


C:\Users\sai\OneDrive\Desktop\DSAL Programs Final\Practical_5.exe
*********MENU*********
1.Insert
2.Display
2.Search
4.Delete
5.Exit
Enter your choice:1
Enter key:A
Enter meaning:Apple
Record Added Successfully
*********MENU*********
1.Insert
2.Display
2.Search
4.Delete
5.Exit
Enter your choice:2
0 :
1 :
2 :
3 :
4 :
5 : A Apple
6 :
7 :
8 :
9 :
*********MENU*********
1.Insert
2.Display
2.Search
4.Delete
5.Exit
Enter your choice:3
Enter key to be searched:A
Key is found Comparisons:1
Key is present
*********MENU*********
1.Insert
2.Display
2.Search
4.Delete
5.Exit
Type here to search
31°C
ENG
IN
8:51 AM
5/23/2023