Experiment 1

INPUT:-

#include<iostream>

#include<string.h>

using namespace std;

struct node {

int value;

node\* next;

};

class hashing {

node\* HashTable[10];

public:

hashing() {

for(int i = 0; i < 10; i++) {

HashTable[i] = NULL;

}

}

int HashFunction(int value) {

return (value % 10);

}

node\* create\_node(int x) {

node\* temp = new node;

temp->next = NULL;

temp->value = x;

return temp;

}

void display() {

for(int i = 0; i < 10; i++) {

node\* temp = HashTable[i];

cout << "a[" << i << "]";

while(temp != NULL) {

cout << " -> " << temp->value;

temp = temp->next;

}

cout << "\n";

}

}

int searchElement(int value) {

int hash\_val = HashFunction(value);

node\* entry = HashTable[hash\_val];

while(entry != NULL) {

if(entry->value == value) {

cout << "\nElement found at: " << hash\_val << " : " << entry->value << endl;

return hash\_val;

}

entry = entry->next;

}

cout << "No element found at key " << hash\_val << ".\n";

return -1;

}

void deleteElement(int value) {

int hash\_val = HashFunction(value);

node\* entry = HashTable[hash\_val];

if(entry == NULL) {

cout << "No Element found at key " << hash\_val << "\n";

return;

}

if(entry->value == value) {

HashTable[hash\_val] = entry->next;

delete entry;

cout << "Element " << value << " deleted from key " << hash\_val << "\n";

return;

}

while(entry->next != NULL && entry->next->value != value) {

entry = entry->next;

}

if(entry->next != NULL) {

node\* temp = entry->next;

entry->next = temp->next;

delete temp;

cout << "Element " << value << " deleted from key " << hash\_val << "\n";

} else {

cout << "No element with value " << value << " found to delete.\n";

}

}

void insertElement(int value) {

int hash\_val = HashFunction(value);

node\* new\_node = create\_node(value);

node\* temp = HashTable[hash\_val];

if(temp == NULL) {

HashTable[hash\_val] = new\_node;

} else {

while(temp->next != NULL) {

temp = temp->next;

}

temp->next = new\_node;

}

cout << "Element " << value << " inserted at key " << hash\_val << "\n";

}

};

int main() {

int ch;

int data, search, del;

hashing h;

do {

cout << "\nTelephone : \n1.Insert\n2.Display\n3.Search\n4.Delete\n5.Exit\n";

cin >> ch;

switch(ch) {

case 1:

cout << "\nEnter phone no. to be inserted: ";

cin >> data;

h.insertElement(data);

break;

case 2:

h.display();

break;

case 3:

cout << "\nEnter the number to be searched: ";

cin >> search;

h.searchElement(search);

break;

case 4:

cout << "\nEnter the phone number to be deleted: ";

cin >> del;

h.deleteElement(del);

break;

case 5:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid choice! Try again.\n";

}

} while(ch != 5);

return 0;

}

OUTPUT:-

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

1

Enter phone no. to be inserted: 1234

Element 1234 inserted at key 4

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

1

Enter phone no. to be inserted: 4567

Element 4567 inserted at key 7

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

1

Enter phone no. to be inserted: 9764

Element 9764 inserted at key 4

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

2

a[0]

a[1]

a[2]

a[3]

a[4] -> 1234 -> 9764

a[5]

a[6]

a[7] -> 4567

a[8]

a[9]

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

3

Enter the number to be searched: 1234

Element found at: 4 : 1234

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

4

Enter the phone number to be deleted: 4567

Element 4567 deleted from key 7

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

2

a[0]

a[1]

a[2]

a[3]

a[4] -> 1234 -> 9764

a[5]

a[6]

a[7]

a[8]

a[9]

Telephone :

1.Insert

2.Display

3.Search

4.Delete

5.Exit

5

Exiting program.

=== Code Execution Successful ===