Name: Syed Muhammad Raza Ali (02-134231-028)

Lab 12: Hashing

Objective(s): Upon completion of this lab session, students will be able to:

Implement hashing table
Implement hashing function
Able to build solutions according to scenarios by applying hashing.

Exercise 1: Student records

Write a program to design a system for storing student's records keyed using last three digits of enrollment numbers and the size of hash table must be a prime number. Resolve the collision with quadratic probing if occurs.

Your program should have following functions

- **HashTable( )** : This function is used to create a new hash table.
- **Delete( )**: This function is used to delete enrollment number from the hash table.
- **Get( )**: This function is used to search enrollment number inside the hash table.
- **Put( ):** This function is used to insert enrollment number inside the hash table.

Code:

```
#include <iostream>
using namespace std;

struct hashTable {
	int key;
	string name;
};

const int tableSize = 11;
hashTable table[tableSize];

void initHashTable() {
	for (int i = 0; i < tableSize; i++) {
		table[i].key = -1;
	}
```

```
}

int hashFunction(int key) {
	return key % tableSize;
}

void insertRecord(int key,string name) {
	int keyIndex = hashFunction(key);
	if (table[keyIndex].key == -1) {
		table[keyIndex].key = key;
		table[keyIndex].name = name;
		cout << "key successfully inserted...\n";
	}
	else {
		int newKeyIndex = keyIndex;		//quadratic collision
handling

		for (int i = 0; i < tableSize; i++) {
			if (table[newKeyIndex].key != -1) {
				newKeyIndex = (keyIndex + i * i) % tableSize;
			}
			else {
				break;
			}
		}
		if (table[newKeyIndex].key == -1) {
			table[newKeyIndex].key = key;
			table[newKeyIndex].name = name;
			cout << "Collision was handled successfully...\n";
		}
	}
}

void searchRecord(int key) {
	cout << "============ Search Key ============\n";
	int keyIndex = hashFunction(key);
	if (table[keyIndex].key != -1) {
		cout << "key : " << table[keyIndex].key << "\nname : " <<
table[keyIndex].name << "\n";
	}
	else {
		cout << "Record doesnt exist...\n";
	}
}

void deleteRecord(int key) {
	cout << "============ Delete Key ============\n";
	int keyIndex = hashFunction(key);
	if (table[keyIndex].key != -1) {
		table[keyIndex].key = -1;
		table[keyIndex].name = "";
		cout << "key was successfully deleted...\n";
	}
```

```
	else {
		cout << "There was an error deleting the key...\n";
	}
}

void displayHashTable() {
	cout << "============ Display Table ============\n";

	for (int i = 0; i < tableSize; i++) {
		cout << i << "  ------   " << table[i].key << "   " <<
table[i].name << " \n";
	}
}

int main() {

	initHashTable();
	displayHashTable();

	insertRecord(281, "raza");
	insertRecord(346, "aimen");
	insertRecord(823, "bilal");
	insertRecord(112, "muskan");
	insertRecord(234, "huzaifa");
	insertRecord(847, "ali");
	insertRecord(387, "sana");
	insertRecord(392, "sara");
	insertRecord(586, "alina");

	displayHashTable();

	searchRecord(112);
	displayHashTable();

	deleteRecord(281);
	displayHashTable();


	return 0;
}
```

Output:

```
key successfully inserted...
key successfully inserted...
key successfully inserted...
key successfully inserted...
key successfully inserted...
key successfully inserted...
Collision was handled successfully...
Collision was handled successfully...
Collision was handled successfully...
============ Display Table ============
0  ------   847  ali
1  ------   -1
2  ------   112  muskan
3  ------   234  huzaifa
4  ------   586  alina
5  ------   346  aimen
6  ------   281  raza
7  ------   387  sana
8  ------   392  sara
9  ------   823  bilal
10  ------   -1
============ Search Key ============
key : 112
name : muskan
============ Display Table ============
0  ------   847  ali
1  ------   -1
2  ------   112  muskan
3  ------   234  huzaifa
4  ------   586  alina
5  ------   346  aimen
6  ------   281  raza
7  ------   387  sana
8  ------   392  sara
9  ------   823  bilal
10  ------   -1
============ Delete Key ============
key was successfully deleted...
============ Display Table ============
0  ------   847  ali
1  ------   -1
2  ------   112  muskan
3  ------   234  huzaifa
4  ------   586  alina
5  ------   346  aimen
6  ------   -1
7  ------   387  sana
8  ------   392  sara
9  ------   823  bilal
10  ------   -1
```

Bahria University
Discovering Knowledge

CSL-221 Data structure & Algorithm

Semester 03

Exercise 2: Character codes

Suppose that the following character codes are used: 'A' =1, 'B' =2, ... ,'Y' =25, 'Z' = 26. Using a hash table with eleven locations and the hashing function h(identifier) = average % 11, where average is the average of the codes of the first and last letters in identifier, show the hash table that results when the following identifiers are inserted in the order given, assuming that collisions are resolved using linear probing: BETA, RATE, FREQ, ALPHA, MEAN, SUM, NUM, BAR,WAGE, PAY, KAPPA.

Write a program for the above problem.

Code:

```
#include <iostream>
using namespace std;

struct inputData {
	char character;
	int code;

};
inputData dataSet[26];

struct hashTable {
	string key;
	int code;

};

const int tableSize = 11;
hashTable table[tableSize];

//For Dataset
void initDataset() {
	char character = 'A';
	int code = 1;

	for (int i = 0; i < 26; i++) {
		dataSet[i].character = character;
		dataSet[i].code = code;
		character += 1;
		code += 1;
	}
}
void printDataset() {
	for (int i = 0; i < 26; i++) {
		cout << i << "  ------  " << dataSet[i].character << "   " <<
dataSet[i].code << "\n";
	}
}
```

```
//For hashTable
void initHashTable(){
	for (int i = 0; i < tableSize; i++) {
		table[i].key = "";
	}
}

int  hashFunction(string identifier) {
	char c1 = identifier[0];
	char c2 = identifier[size(identifier) - 1];
	int code1 = 0;
	int code2 = 0;
	for (int i = 0; i < tableSize; i++) {
		if (c1 == dataSet[i].character) {
			code1 = dataSet[i].code;
		}
		else if (c2 == dataSet[i].character) {
			code2 = dataSet[i].code;
		}
	}
	int average = (code1+ code2) / 2;
	return average % tableSize;
}

void insertRecord(string identifier) {
	int keyIndex = hashFunction(identifier);

	if (table[keyIndex].key == "") {
		table[keyIndex].key = identifier;
		cout << "Record was inserted successfully...\n";
	}
	else {
		for (int i = 0; i < tableSize; i++) {
			if (table[keyIndex].key != "") {
				keyIndex = (keyIndex + 1) % tableSize;
			}
		}
		table[keyIndex].key = identifier;
		cout << "Collision was handled through linear probing...\n";
	}
}
void printHashtable(){
	for (int i = 0; i < tableSize; i++) {
		cout << i << "  ------  " << table[i].key<< "\n";
	}
}

int main() {

	initDataset();
	printDataset();
	initHashTable();
```

```
insertRecord("BETA");
insertRecord("RATE");
insertRecord("FREQ");
insertRecord("ALPHA");
insertRecord("MEAN");
insertRecord("SUM");
insertRecord("NUM");
insertRecord("BAR");
insertRecord("WAGE");
insertRecord("PAY");
insertRecord("KAPPA");
printHashtable();
```

```
        return 0;
}
```

```
0  ------  A  1
1  ------  B  2
2  ------  C  3
3  ------  D  4
4  ------  E  5
5  ------  F  6
6  ------  G  7
7  ------  H  8
8  ------  I  9
9  ------  J  10
10  ------  K  11
11  ------  L  12
12  ------  M  13
13  ------  N  14
14  ------  O  15
15  ------  P  16
16  ------  Q  17
17  ------  R  18
18  ------  S  19
19  ------  T  20
20  ------  U  21
21  ------  V  22
22  ------  W  23
23  ------  X  24
24  ------  Y  25
25  ------  Z  26
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
Collision was handled through linear probing...
Collision was handled through linear probing...
Collision was handled through linear probing...
Collision was handled through linear probing...
Collision was handled through linear probing...
Collision was handled through linear probing...
Collision was handled through linear probing...
0  ------  ALPHA
1  ------  BETA
2  ------  RATE
3  ------  FREQ
4  ------  MEAN
5  ------  SUM
6  ------  NUM
7  ------  BAR
8  ------  WAGE
9  ------  PAY
10  ------  KAPPA
```

Exercise 3: Country's capital

As a software engineer at **GeoMap Solutions,** you are tasked to design a system that stores country names in a hash table according to their lengths. Write a program and perform following task:

1. Insert given keys in the hashtable: United States, Brazil, Japan, Germany, Pakistan, Argentina, Australia.
2. Delete a country from the hastable.
3. Update a capital using its country name.
4. Search for capital with respect to its country name.

Note: Use linear probing for collision handling if there's any.

Code:

```
#include <iostream>
using namespace std;

struct hashTable {
	string countryName;
	string capital;
};

const int tableSize = 14;
hashTable table[tableSize];

void initHashTable() {
	for (int i = 0; i < tableSize; i++) {
		table[i].countryName = "";
	}
}

int hashFunction(string countryName) {
	return size(countryName) % tableSize;
}

void insertRecord(string countryName,string capital) {
	int keyIndex = hashFunction(countryName);

	if (table[keyIndex].countryName == "") {
		table[keyIndex].countryName = countryName;
		table[keyIndex].capital = capital;
		cout << "Record was inserted successfully...\n";
	}
	else {
		for (int i = 0; i < tableSize; i++) {
			if (table[keyIndex].countryName != "") {
```

```
                        keyIndex = (keyIndex + 1) % tableSize;
                }
            }
            table[keyIndex].countryName = countryName;
            table[keyIndex].capital = capital;
            cout << "collision was handled through linear probing...\n";
        }
}
void searchRecord(string countryName) {
        int keyIndex = hashFunction(countryName);
        if(countryName == table[keyIndex].countryName and
table[keyIndex].countryName!="")
        cout << "Country name : " << table[keyIndex].countryName<<"\nCapital :
"<<table[keyIndex].capital<<"\n";
}
void deleteRecord(string countryName) {
        int keyIndex = hashFunction(countryName);
        if (countryName == table[keyIndex].countryName and
table[keyIndex].countryName != "") {
            table[keyIndex].countryName = "";
            table[keyIndex].capital= "";
            cout << "Record has been successfully deleted...\n";
        }
}
void updateRecord(string countryName,string newCapital) {
        int keyIndex = hashFunction(countryName);
        if (countryName == table[keyIndex].countryName and
table[keyIndex].countryName != "") {
            table[keyIndex].capital = newCapital;
            cout << "Capital has been successfully updated...\n";
        }
}
void displayHashTable() {
        for (int i = 0; i < tableSize; i++) {
            cout << i << "  ------  " << table[i].countryName << "    " <<
table[i].capital << "\n";
        }
}

int main() {

        initHashTable();
        insertRecord("United States","Washington DC");
        insertRecord("Brazil", "Brasilia");
        insertRecord("Japan", "Tokyo");
        insertRecord("Germany", "Berlin");
        insertRecord("Pakistan", "Islamabad");
        insertRecord("Argentina", "Buenos");
        insertRecord("Australia", "Canberra");

        displayHashTable();
        cout << "====== Search Record ======\n";
        searchRecord("United States");
```

```
        searchRecord("Pakistan");
        cout << "====== Delete Record ======\n";
        searchRecord("Japan");
        searchRecord("Argentina");
        cout << "====== Update Record ======\n";
        updateRecord("Pakistan","Karachi");

        displayHashTable();
        return 0;
}
```

Output:

```
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
Record was inserted successfully...
collision was handled through linear probing...
0  ------
1  ------
2  ------
3  ------
4  ------
5  ------  Japan    Tokyo
6  ------  Brazil    Brasilia
7  ------  Germany    Berlin
8  ------  Pakistan    Islamabad
9  ------  Argentina    Buenos
10  ------  Australia    Canberra
11  ------
12  ------
13  ------  United States    Washington DC
====== Search Record ======
Country name : United States
Capital : Washington DC
Country name : Pakistan
Capital : Islamabad
====== Delete Record ======
Country name : Japan
Capital : Tokyo
Country name : Argentina
Capital : Buenos
====== Update Record ======
Capital has been successfully updated...
0  ------
1  ------
2  ------
3  ------
4  ------
5  ------  Japan    Tokyo
6  ------  Brazil    Brasilia
7  ------  Germany    Berlin
8  ------  Pakistan    Karachi
9  ------  Argentina    Buenos
10  ------  Australia    Canberra
11  ------
12  ------
13  ------  United States    Washington DC
```