Practical 9

Aim:

Use Hashing Techniques to store the records. Use any two methods.

Code:

```
#include <iostream>
#include <list>
#include <vector>
using namespace std;
class HashTableSeparateChaining {
private:
	vector<list<pair<int, string>>> table;
	int capacity;
	int hashFunction(int key) {
		return key % capacity;
	}
public:
	HashTableSeparateChaining(int size) : capacity(size) {
		table.resize(size);
	}
	void insert(int key, const string& value) {
		int index = hashFunction(key);
		table[index].push_back({ key, value });
	}
	void search(int key) {
		int index = hashFunction(key);
		for (const auto& entry : table[index]) {
			if (entry.first == key) {
				cout << "Key: " << entry.first << ", Value: " << entry.second << endl;
				return;
			}
		}
		cout << "Key not found." << endl;
	}
	void remove(int key) {
		int index = hashFunction(key);
		auto& entries = table[index];
		for (auto it = entries.begin(); it != entries.end(); ++it) {
			if (it->first == key) {
				entries.erase(it);
				cout << " Key removed !!" << endl;
				return;
			}
		}
		cout << "Key not found." << endl;
```

Practical 9

```
        }
};
int main() {
        int size;
        cout << "Enter the size of the hash table: ";
        cin >> size;
        HashTableSeparateChaining ht(size);
        int key;
        string value;
        int numRecords;
        cout << "Enter the number of records: ";
        cin >> numRecords;
        for (int i = 0; i < numRecords; ++i) {
                cout << "Enter key and value for record " << i + 1 << ": ";
                cin >> key >> value;
                ht.insert(key, value);
        }
        int searchKey;
        cout << "Enter the key to search: ";
        cin >> searchKey;
        ht.search(searchKey);
        int removeKey;
        cout << "Enter the key to remove: ";
        cin >> removeKey;
        ht.remove(removeKey);
        return 0;

}
```

Output: