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

#define TABLE_SIZE 5

// ---------------------- COMMON FUNCTIONS ----------------------

// Hash function: calculates index for a username
int hashFunction(string username) {
    int sum = 0;
    for (char c : username)
        sum += c;
    return sum % TABLE_SIZE;
}

// Simple password hashing: converts each char to (ASCII % 10) and concatenates
string hashPassword(string password) {
    string hashed = "";
    for (char c : password)
        hashed += to_string((int)c % 10);
    return hashed;
}

// ---------------------- CHAINING METHOD ----------------------

struct Node {
    string username;
    string hashedPassword;
    Node* next;
};
```

```
class HashTableChaining {
    Node* table[TABLE_SIZE];

public:
    // Initialize hash table with NULL
    HashTableChaining() {
        for (int i = 0; i < TABLE_SIZE; i++)
            table[i] = NULL;
    }

    // Insert username and password into hash table using chaining
    void insert(string username, string password) {
        int index = hashFunction(username);
        string hashedPass = hashPassword(password);

        Node* newNode = new Node{username, hashedPass, NULL};

        if (table[index] == NULL) {
            table[index] = newNode; // No collision
            cout << "Inserted directly at index " << index << endl;
        } else {
            Node* temp = table[index];
            while (temp->next != NULL)
                temp = temp->next;
            temp->next = newNode; // Collision resolved by chaining
            cout << "Collision resolved by CHAINING at index " << index << endl;
        }
    }

    // Display the hash table with linked lists
    void display() {
        cout << "\n--- Hash Table (Chaining) ---\n";
        for (int i = 0; i < TABLE_SIZE; i++) {
            cout << "Index " << i << ": ";
            Node* temp = table[i];
            if (!temp)
                cout << "Empty";
            while (temp) {
                cout << "[" << temp->username << " : " << temp->hashedPassword << "] -> ";
                temp = temp->next;
            }
            cout << "NULL\n";
        }
    }
};

// ---------------------- OPEN ADDRESSING METHOD ----------------------

class HashTableOpenAddressing {
    string user[TABLE_SIZE];
    string pass[TABLE_SIZE];
    bool used[TABLE_SIZE];

public:
    // Initialize hash table and used flags
    HashTableOpenAddressing() {
        for (int i = 0; i < TABLE_SIZE; i++)
            used[i] = false;
    }

    // Insert username and password using linear probing
    void insert(string username, string password) {
```

```
    int idx = hashFunction(username);
    int start = idx;
    string hashedPass = hashPassword(password);

    // Find next free slot if collision occurs
    while (used[idx]) {
        idx = (idx + 1) % TABLE_SIZE;
        if (idx == start) {
            cout << "Table full, cannot insert!\n";
            return;
        }
    }

    user[idx] = username;
    pass[idx] = hashedPass;
    used[idx] = true;
    cout << "Inserted successfully at index " << idx << endl;
}

// Display the hash table
void display() {
    cout << "\n--- Hash Table (Open Addressing) ---\n";
    for (int i = 0; i < TABLE_SIZE; i++) {
        cout << "Index " << i << ": ";
        if (used[i])
            cout << "[" << user[i] << " : " << pass[i] << "]";
        else
            cout << "Empty";
        cout << "\n";
    }
}
```

```cpp
};

// ---------------------- MAIN FUNCTION ----------------------

int main() {
    int choice;
    cout << "--------------------------\n";
    cout << "HASHING TECHNIQUES DEMO\n";
    cout << "--------------------------\n";
    cout << "1. Chaining\n";
    cout << "2. Open Addressing (Linear Probing)\n";
    cout << "Enter choice: ";
    cin >> choice;

    int n;
    cout << "\nEnter number of users: ";
    cin >> n;

    if (choice == 1) {
        HashTableChaining ht;
        for (int i = 0; i < n; i++) {
            string username, password;
            cout << "\nEnter username: ";
            cin >> username;
            cout << "Enter password: ";
            cin >> password;
            ht.insert(username, password);
        }
        ht.display();
    }
    else if (choice == 2) {
        HashTableOpenAddressing ht;
        for (int i = 0; i < n; i++) {
            string username, password;
            cout << "\nEnter username: ";
            cin >> username;
            cout << "Enter password: ";
            cin >> password;
            ht.insert(username, password);
        }
        ht.display();
    }
    else {
        cout << "Invalid choice!";
    }

    return 0;
}
```

```
Enter username: user1
Enter password: pict123
✅ Inserted directly at index 1

Enter username: user2
Enter password: BC98
✅ Inserted directly at index 2
```

```
Enter username: test1

Enter password: hello

⚠ Collision resolved by CHAINING at index 2
```