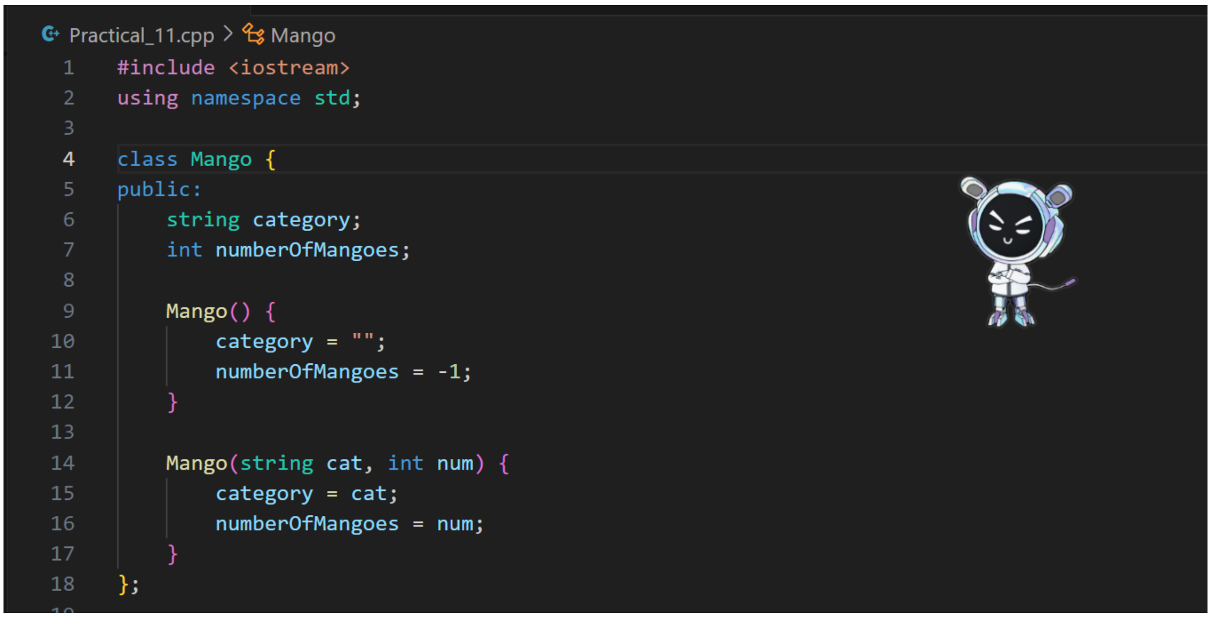
Practical 11

NAME: Omkar Londhe

PRN: 202301040027 BATCH: B4

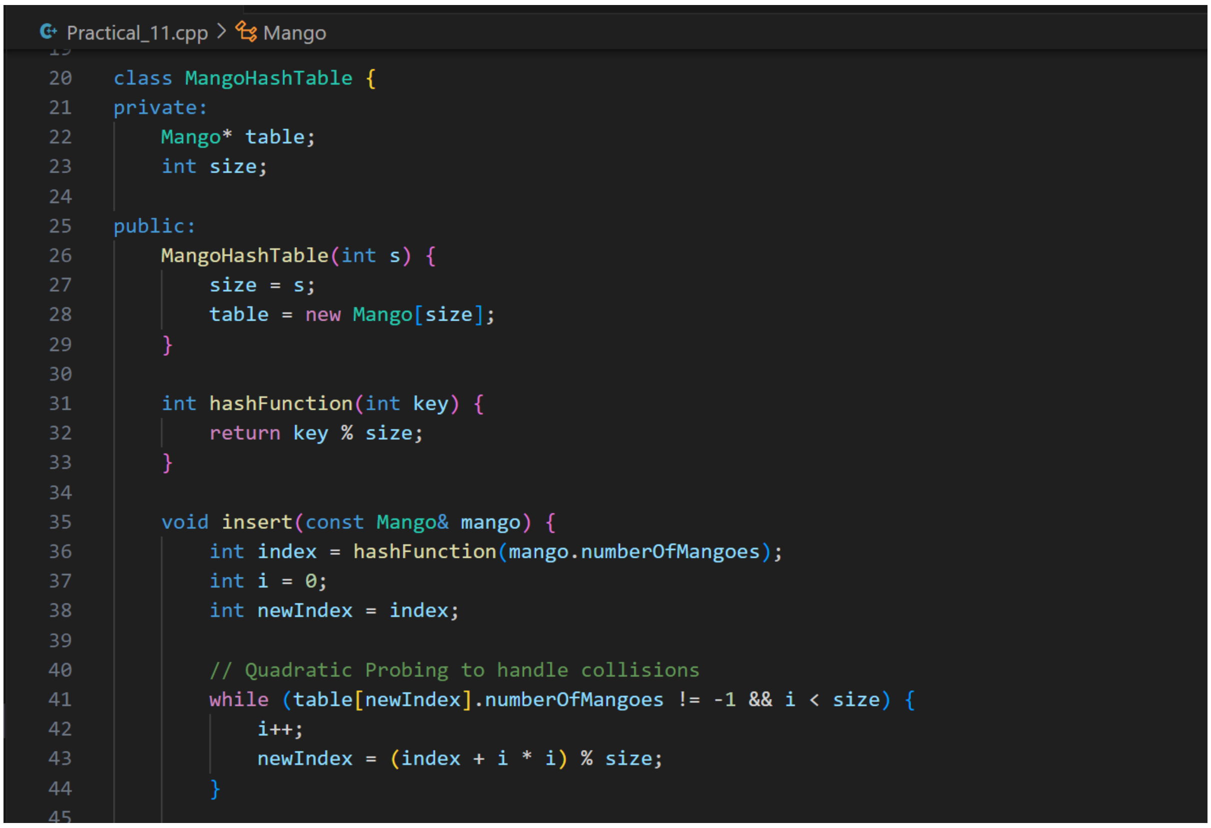
ROLL NO: 59

Code:

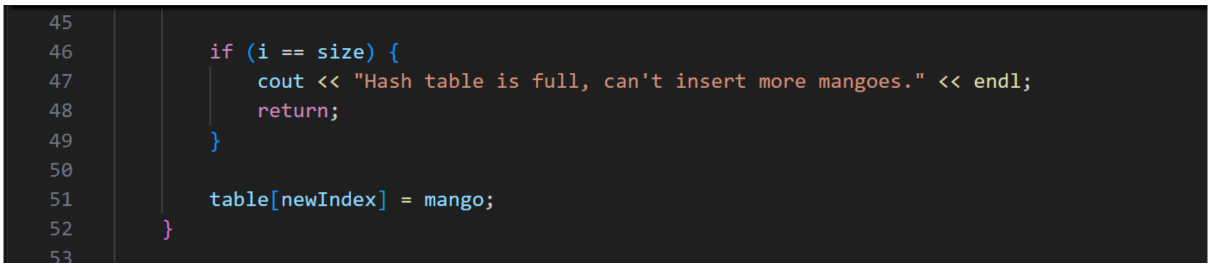


Practical_11.cpp > Mango

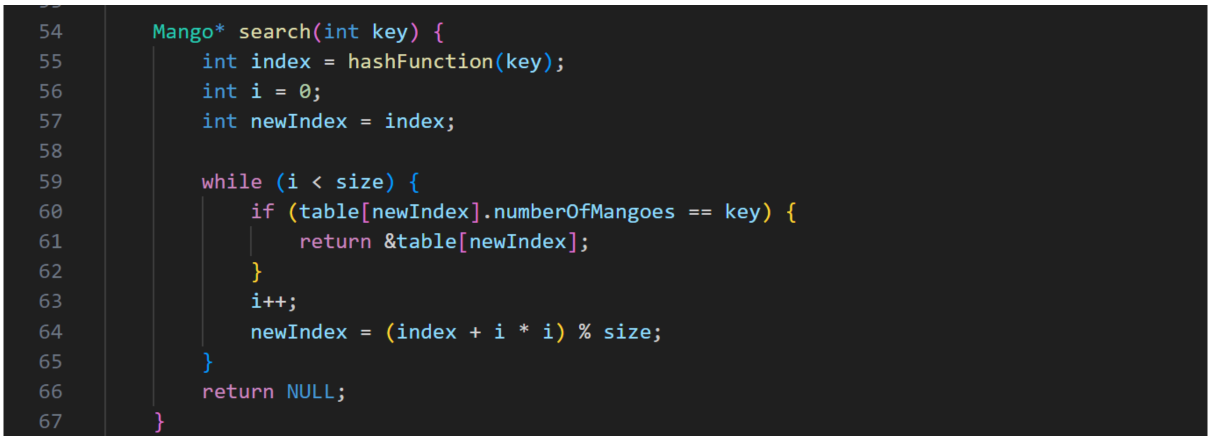
```cpp
#include <iostream>
using namespace std;

class Mango {
public:
    string category;
    int numberOfMangoes;

    Mango() {
        category = "";
        numberOfMangoes = -1;
    }

    Mango(string cat, int num) {
        category = cat;
        numberOfMangoes = num;
    }
};
```



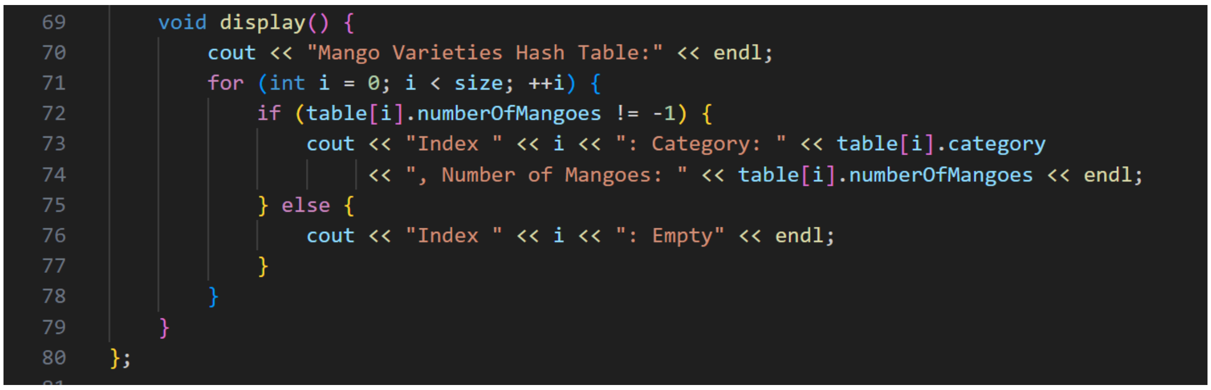
```cpp
class MangoHashTable {
private:
    Mango* table;
    int size;

public:
    MangoHashTable(int s) {
        size = s;
        table = new Mango[size];
    }

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

    void insert(const Mango& mango) {
        int index = hashFunction(mango.numberOfMangoes);
        int i = 0;
        int newIndex = index;

        // Quadratic Probing to handle collisions
        while (table[newIndex].numberOfMangoes != -1 && i < size) {
            i++;
            newIndex = (index + i * i) % size;
        }
```



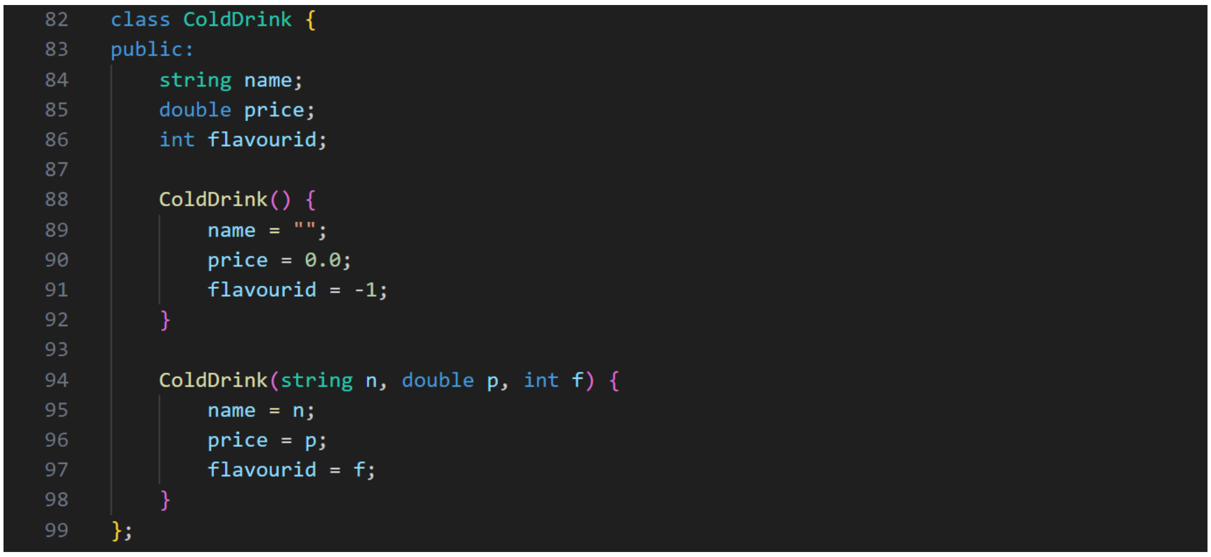
```cpp

        if (i == size) {
            cout << "Hash table is full, can't insert more mangoes." << endl;
            return;
        }

        table[newIndex] = mango;
    }
```



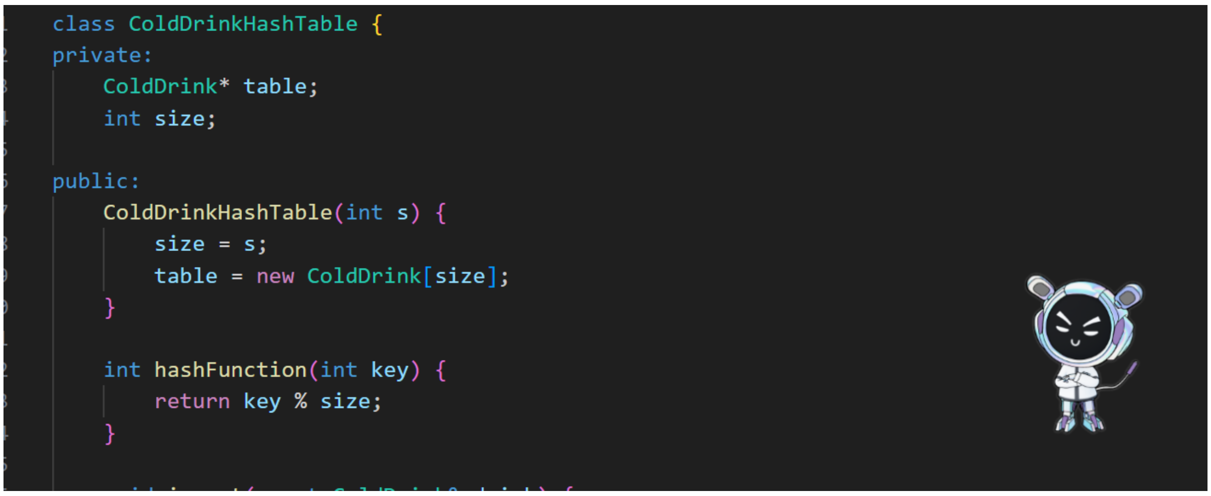
```cpp
    Mango* search(int key) {
        int index = hashFunction(key);
        int i = 0;
        int newIndex = index;

        while (i < size) {
            if (table[newIndex].numberOfMangoes == key) {
                return &table[newIndex];
            }
            i++;
            newIndex = (index + i * i) % size;
        }
        return NULL;
    }
```



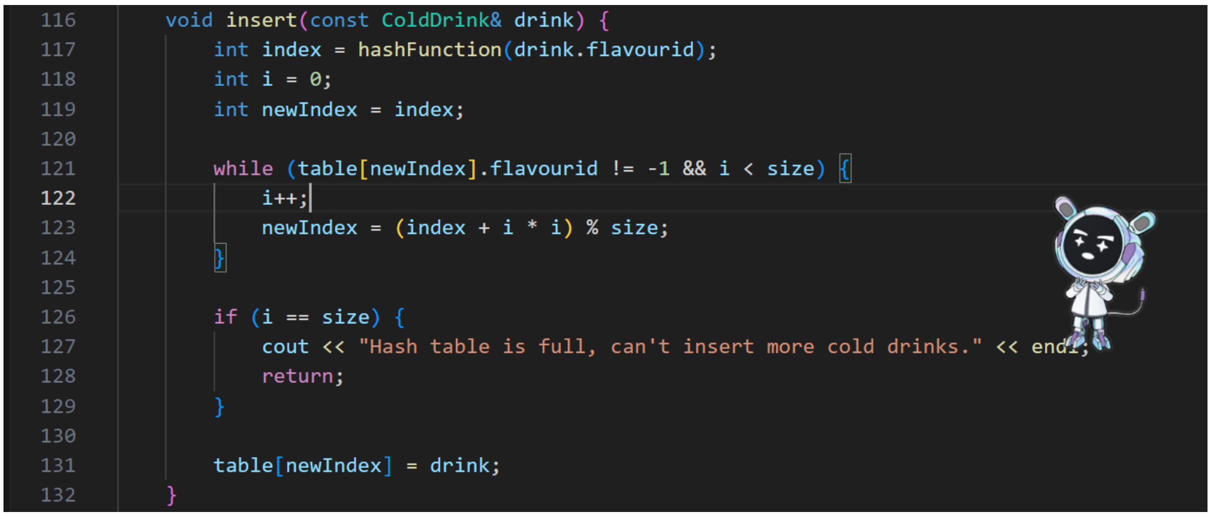
```cpp
    void display() {
        cout << "Mango Varieties Hash Table:" << endl;
        for (int i = 0; i < size; ++i) {
            if (table[i].numberOfMangoes != -1) {
                cout << "Index " << i << ": Category: " << table[i].category
                     << ", Number of Mangoes: " << table[i].numberOfMangoes << endl;
            } else {
                cout << "Index " << i << ": Empty" << endl;
            }
        }
    }
};
```



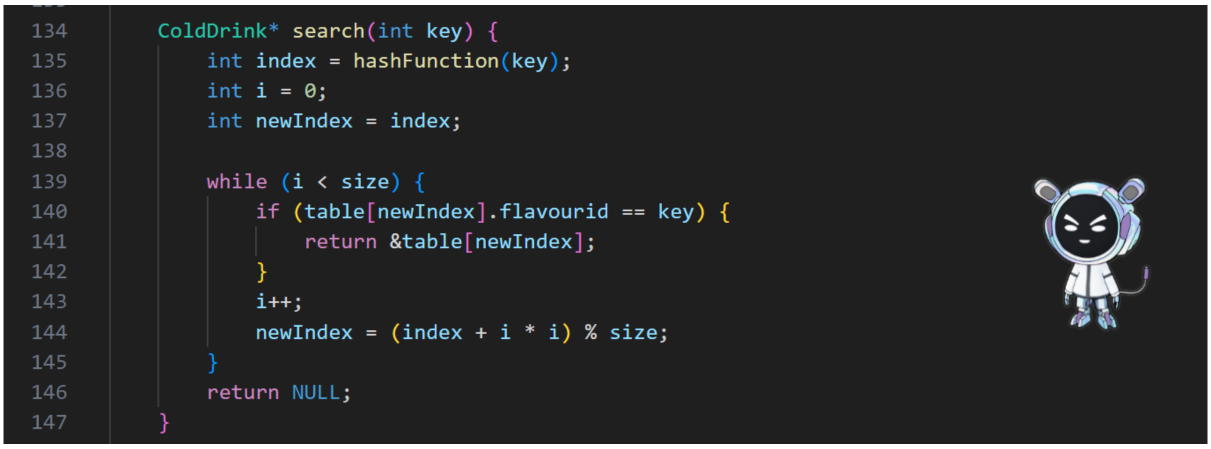
```cpp
class ColdDrink {
public:
    string name;
    double price;
    int flavourid;

    ColdDrink() {
        name = "";
        price = 0.0;
        flavourid = -1;
    }

    ColdDrink(string n, double p, int f) {
        name = n;
        price = p;
        flavourid = f;
    }
};
```



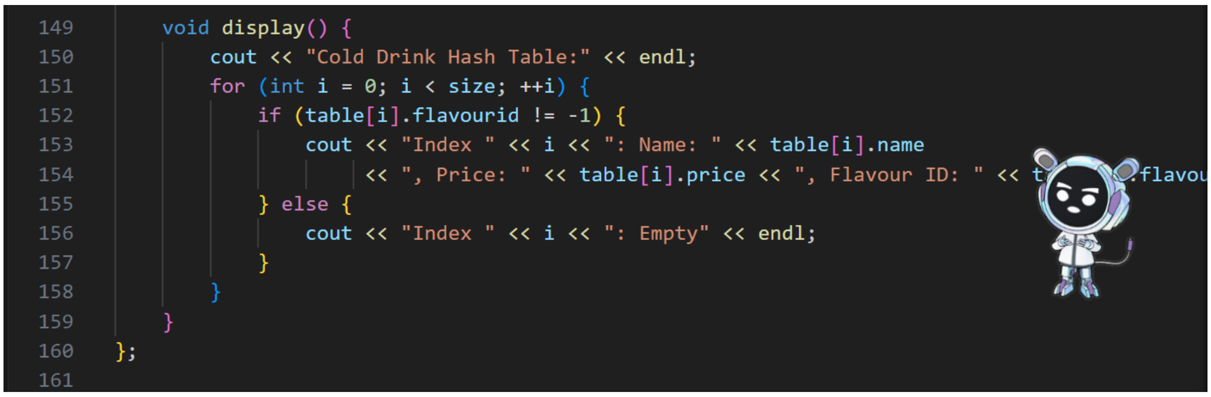
```cpp
class ColdDrinkHashTable {
private:
    ColdDrink* table;
    int size;

public:
    ColdDrinkHashTable(int s) {
        size = s;
        table = new ColdDrink[size];
    }

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

    void insert(const ColdDrink& drink) {
```



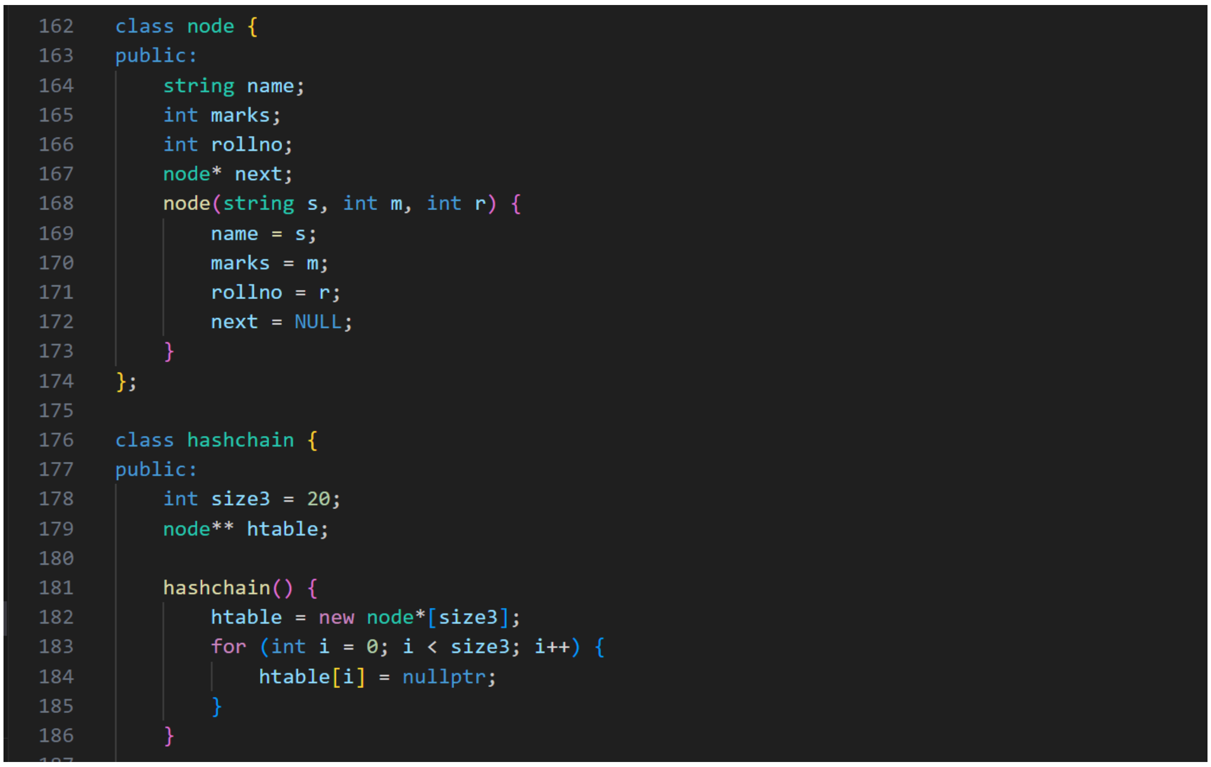
```cpp
    void insert(const ColdDrink& drink) {
        int index = hashFunction(drink.flavourid);
        int i = 0;
        int newIndex = index;

        while (table[newIndex].flavourid != -1 && i < size) {
            i++;
            newIndex = (index + i * i) % size;
        }

        if (i == size) {
            cout << "Hash table is full, can't insert more cold drinks." << endl;
            return;
        }

        table[newIndex] = drink;
    }
```



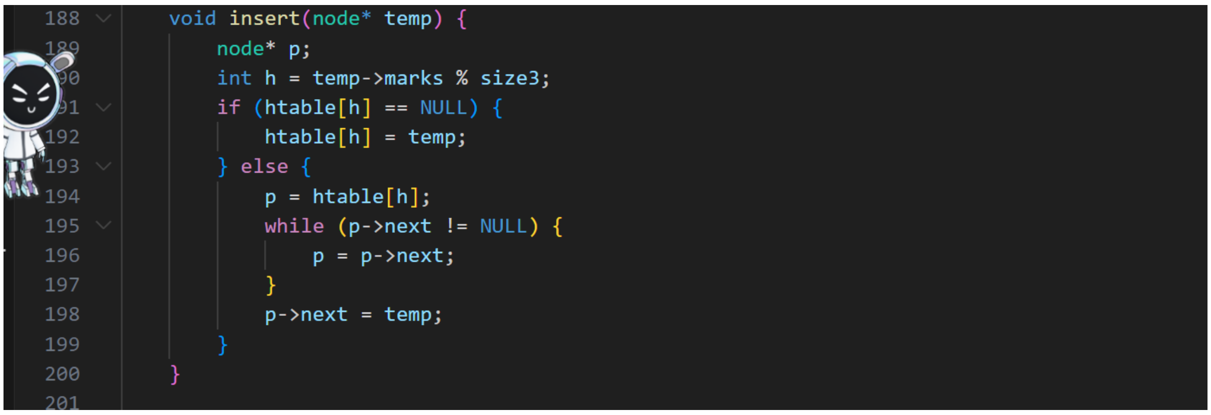
```cpp
    ColdDrink* search(int key) {
        int index = hashFunction(key);
        int i = 0;
        int newIndex = index;

        while (i < size) {
            if (table[newIndex].flavourid == key) {
                return &table[newIndex];
            }
            i++;
            newIndex = (index + i * i) % size;
        }
        return NULL;
    }
```



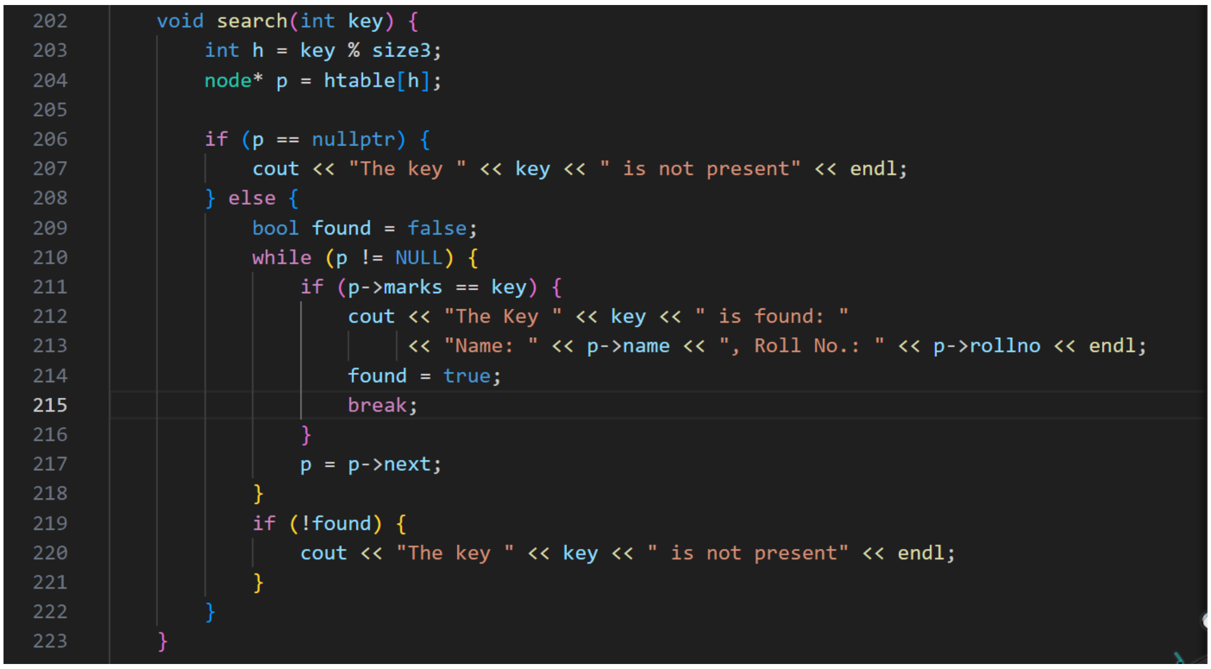
```cpp
    void display() {
        cout << "Cold Drink Hash Table:" << endl;
        for (int i = 0; i < size; ++i) {
            if (table[i].flavourid != -1) {
                cout << "Index " << i << ": Name: " << table[i].name
                     << ", Price: " << table[i].price << ", Flavour ID: " << t[illegible].flavou
            } else {
                cout << "Index " << i << ": Empty" << endl;
            }
        }
    }
};
```



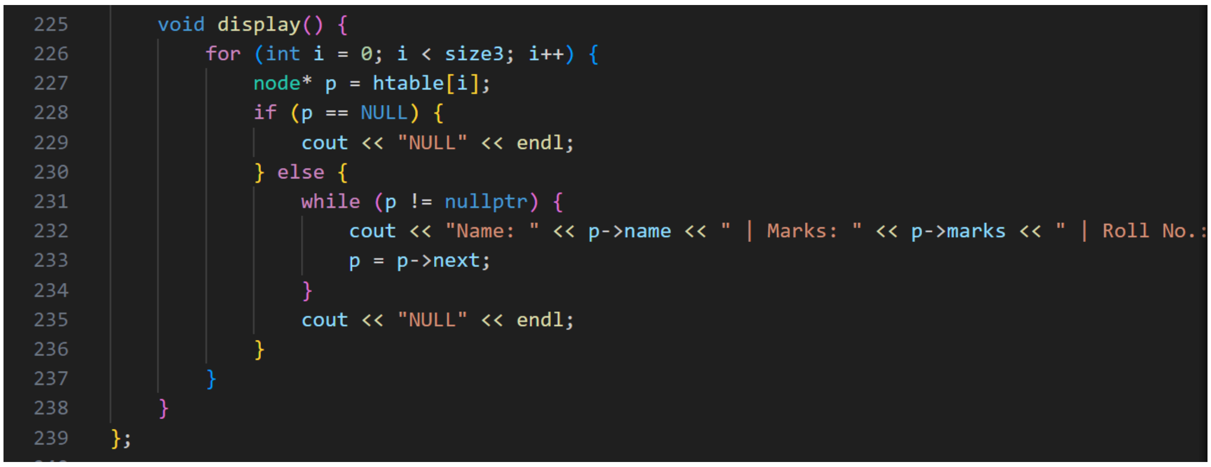
```cpp
class node {
public:
    string name;
    int marks;
    int rollno;
    node* next;
    node(string s, int m, int r) {
        name = s;
        marks = m;
        rollno = r;
        next = NULL;
    }
};

class hashchain {
public:
    int size3 = 20;
    node** htable;

    hashchain() {
        htable = new node*[size3];
        for (int i = 0; i < size3; i++) {
            htable[i] = nullptr;
        }
    }
```



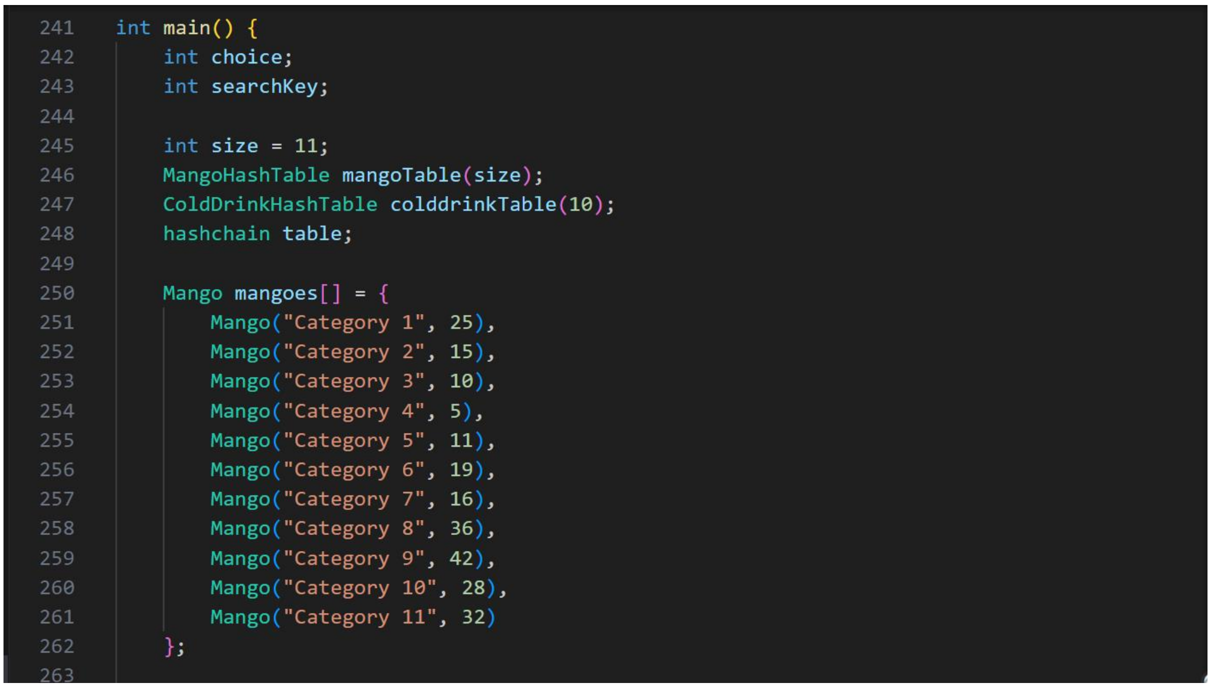
```cpp
    void insert(node* temp) {
        node* p;
        int h = temp->marks % size3;
        if (htable[h] == NULL) {
            htable[h] = temp;
        } else {
            p = htable[h];
            while (p->next != NULL) {
                p = p->next;
            }
            p->next = temp;
        }
    }
```



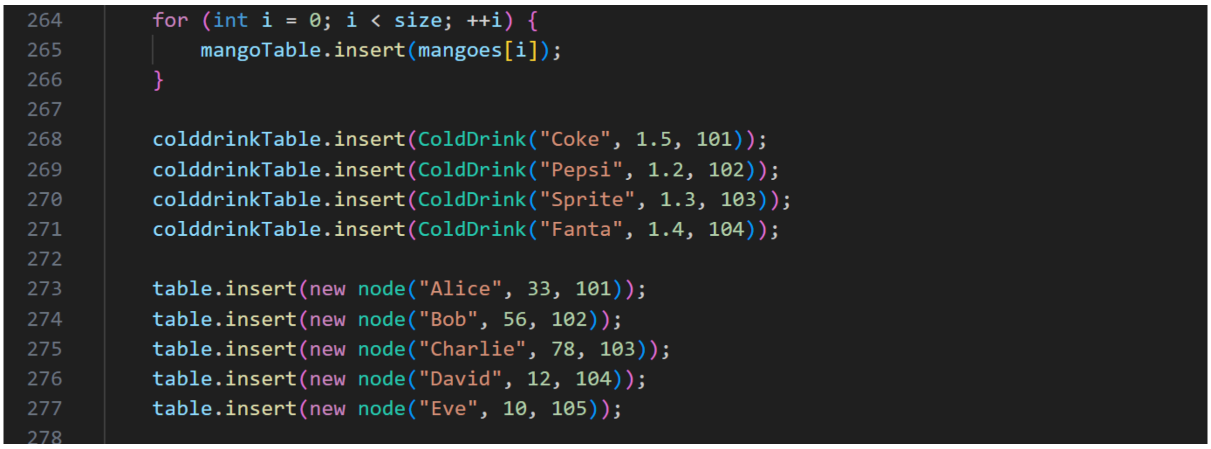
```cpp
    void search(int key) {
        int h = key % size3;
        node* p = htable[h];

        if (p == nullptr) {
            cout << "The key " << key << " is not present" << endl;
        } else {
            bool found = false;
            while (p != NULL) {
                if (p->marks == key) {
                    cout << "The Key " << key << " is found: "
                        << "Name: " << p->name << ", Roll No.: " << p->rollno << endl;
                    found = true;
                    break;
                }
                p = p->next;
            }
            if (!found) {
                cout << "The key " << key << " is not present" << endl;
            }
        }
    }
```



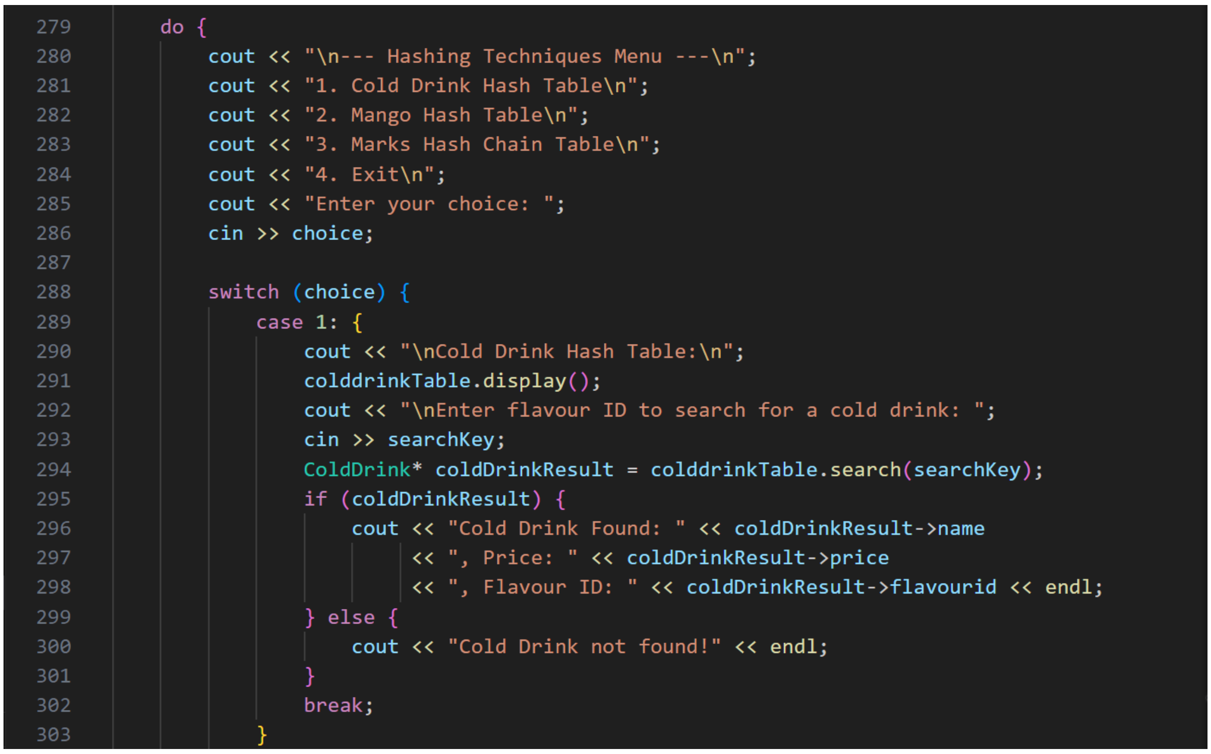
```cpp
    void display() {
        for (int i = 0; i < size3; i++) {
            node* p = htable[i];
            if (p == NULL) {
                cout << "NULL" << endl;
            } else {
                while (p != nullptr) {
                    cout << "Name: " << p->name << " | Marks: " << p->marks << " | Roll No.:
                    p = p->next;
                }
                cout << "NULL" << endl;
            }
        }
    }
};
```



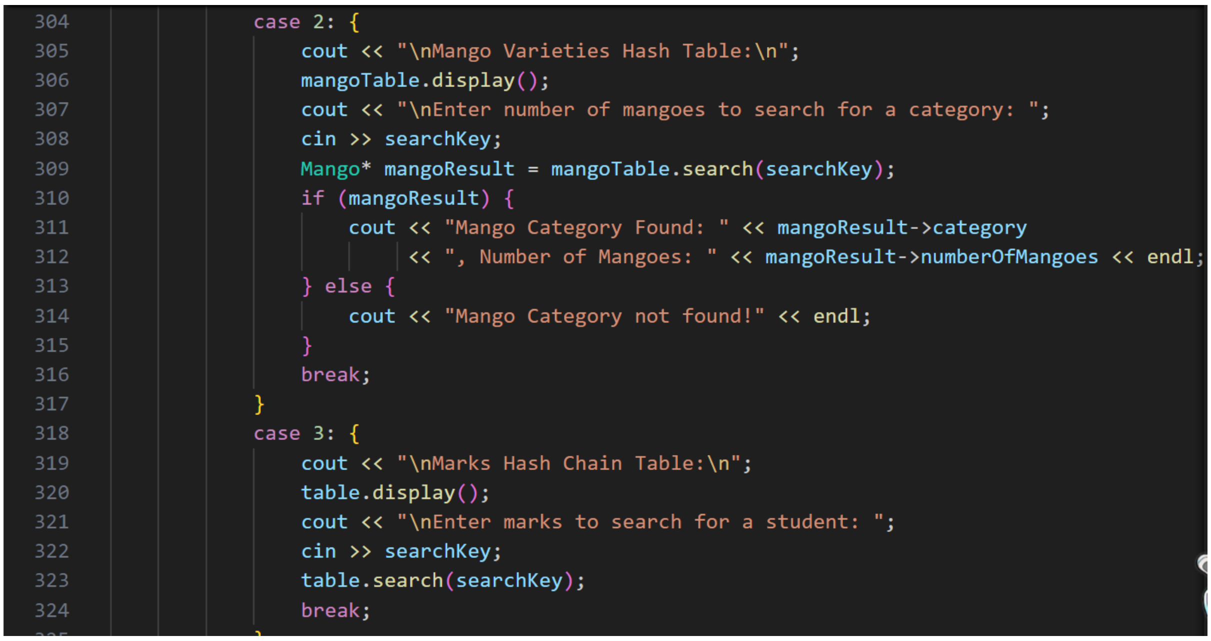
```cpp
int main() {
    int choice;
    int searchKey;

    int size = 11;
    MangoHashTable mangoTable(size);
    ColdDrinkHashTable colddrinkTable(10);
    hashchain table;

    Mango mangoes[] = {
        Mango("Category 1", 25),
        Mango("Category 2", 15),
        Mango("Category 3", 10),
        Mango("Category 4", 5),
        Mango("Category 5", 11),
        Mango("Category 6", 19),
        Mango("Category 7", 16),
        Mango("Category 8", 36),
        Mango("Category 9", 42),
        Mango("Category 10", 28),
        Mango("Category 11", 32)
    };

```



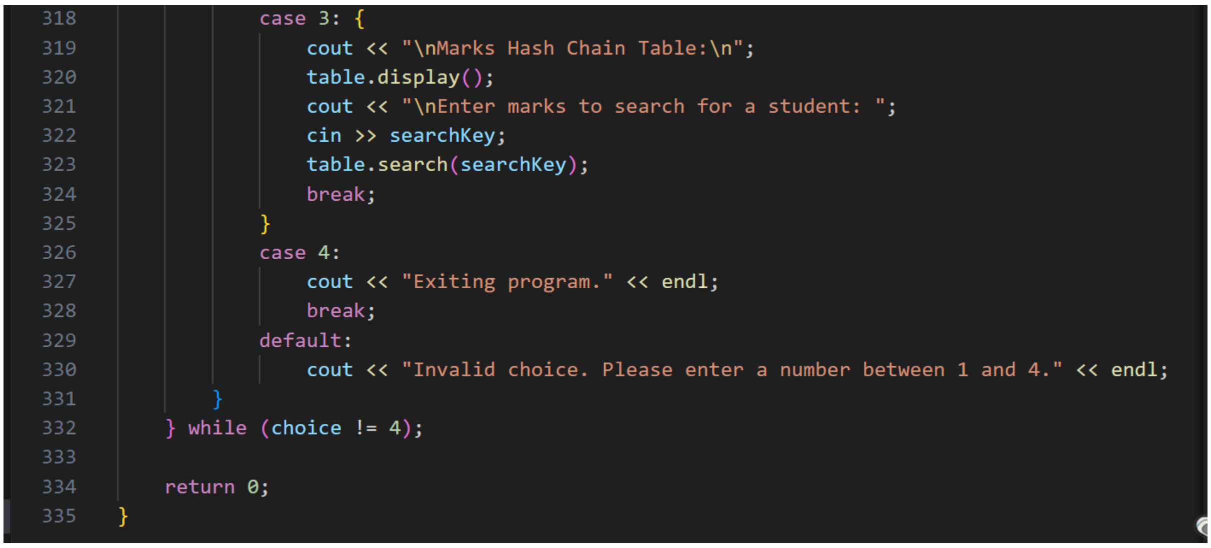
```cpp
    for (int i = 0; i < size; ++i) {
        mangoTable.insert(mangoes[i]);
    }

    colddrinkTable.insert(ColdDrink("Coke", 1.5, 101));
    colddrinkTable.insert(ColdDrink("Pepsi", 1.2, 102));
    colddrinkTable.insert(ColdDrink("Sprite", 1.3, 103));
    colddrinkTable.insert(ColdDrink("Fanta", 1.4, 104));

    table.insert(new node("Alice", 33, 101));
    table.insert(new node("Bob", 56, 102));
    table.insert(new node("Charlie", 78, 103));
    table.insert(new node("David", 12, 104));
    table.insert(new node("Eve", 10, 105));

```



```cpp
    do {
        cout << "\n--- Hashing Techniques Menu ---\n";
        cout << "1. Cold Drink Hash Table\n";
        cout << "2. Mango Hash Table\n";
        cout << "3. Marks Hash Chain Table\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                cout << "\nCold Drink Hash Table:\n";
                colddrinkTable.display();
                cout << "\nEnter flavour ID to search for a cold drink: ";
                cin >> searchKey;
                ColdDrink* coldDrinkResult = colddrinkTable.search(searchKey);
                if (coldDrinkResult) {
                    cout << "Cold Drink Found: " << coldDrinkResult->name
                         << ", Price: " << coldDrinkResult->price
                         << ", Flavour ID: " << coldDrinkResult->flavourid << endl;
                } else {
                    cout << "Cold Drink not found!" << endl;
                }
                break;
            }
```

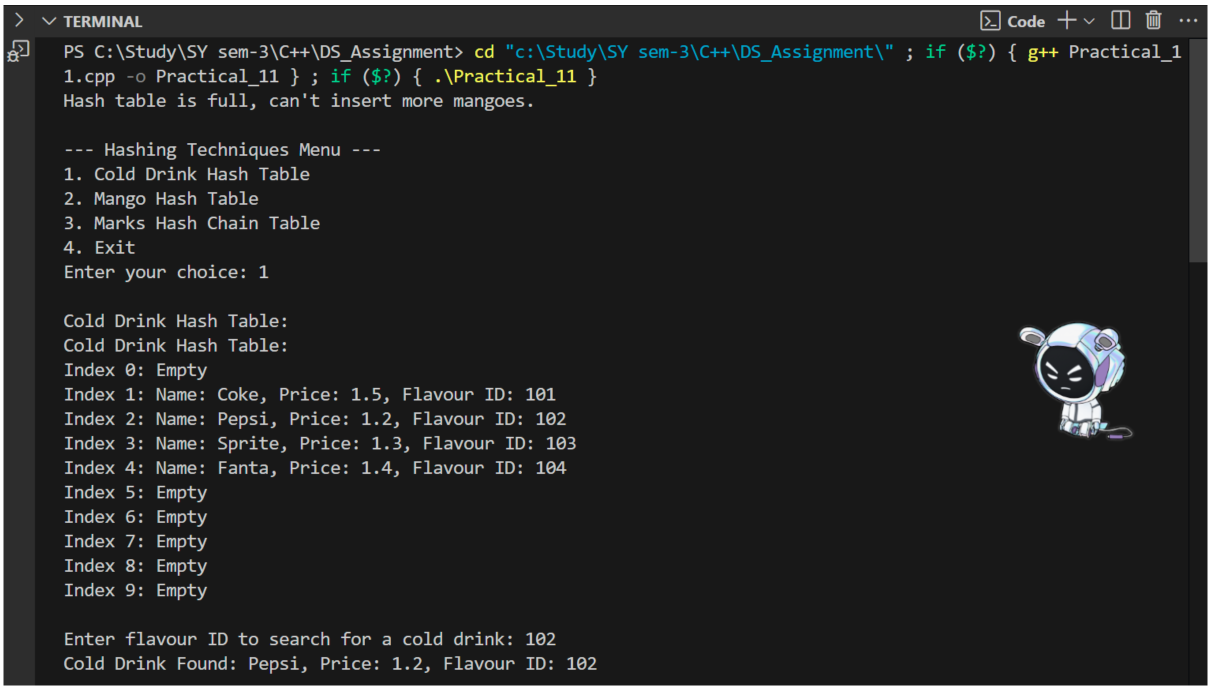


```cpp
            case 2: {
                cout << "\nMango Varieties Hash Table:\n";
                mangoTable.display();
                cout << "\nEnter number of mangoes to search for a category: ";
                cin >> searchKey;
                Mango* mangoResult = mangoTable.search(searchKey);
                if (mangoResult) {
                    cout << "Mango Category Found: " << mangoResult->category
                         << ", Number of Mangoes: " << mangoResult->numberOfMangoes << endl;
                } else {
                    cout << "Mango Category not found!" << endl;
                }
                break;
            }
            case 3: {
                cout << "\nMarks Hash Chain Table:\n";
                table.display();
                cout << "\nEnter marks to search for a student: ";
                cin >> searchKey;
                table.search(searchKey);
                break;
```

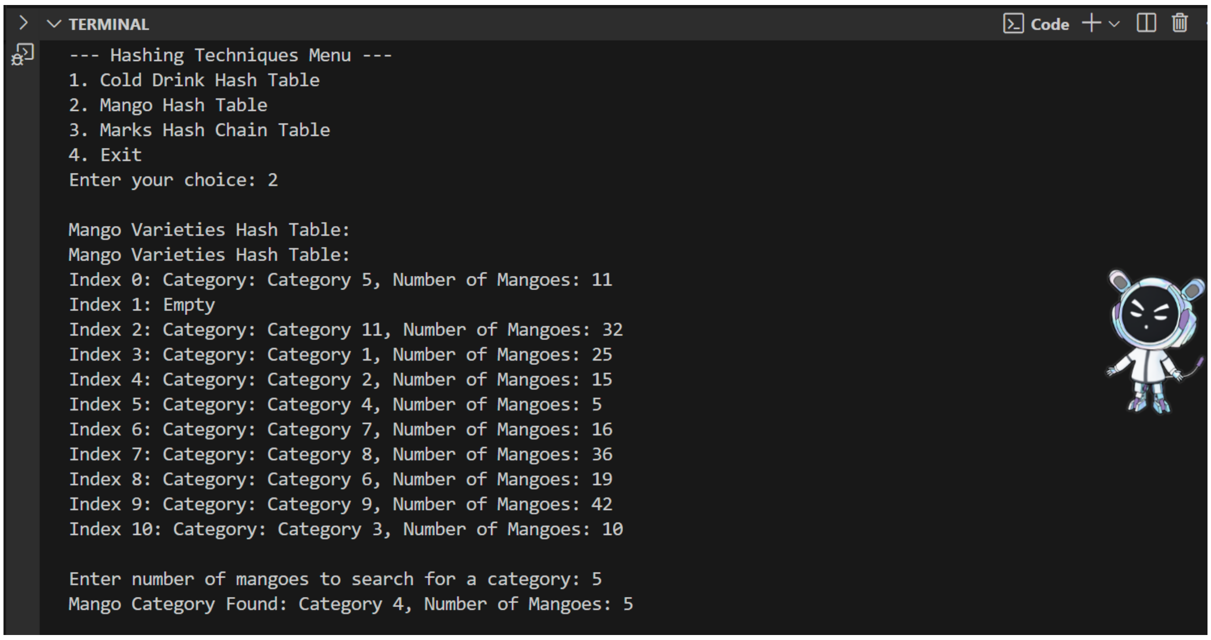


```cpp
                case 3: {
                    cout << "\nMarks Hash Chain Table:\n";
                    table.display();
                    cout << "\nEnter marks to search for a student: ";
                    cin >> searchKey;
                    table.search(searchKey);
                    break;
                }
                case 4:
                    cout << "Exiting program." << endl;
                    break;
                default:
                    cout << "Invalid choice. Please enter a number between 1 and 4." << endl;
            }
        } while (choice != 4);

        return 0;
    }
```

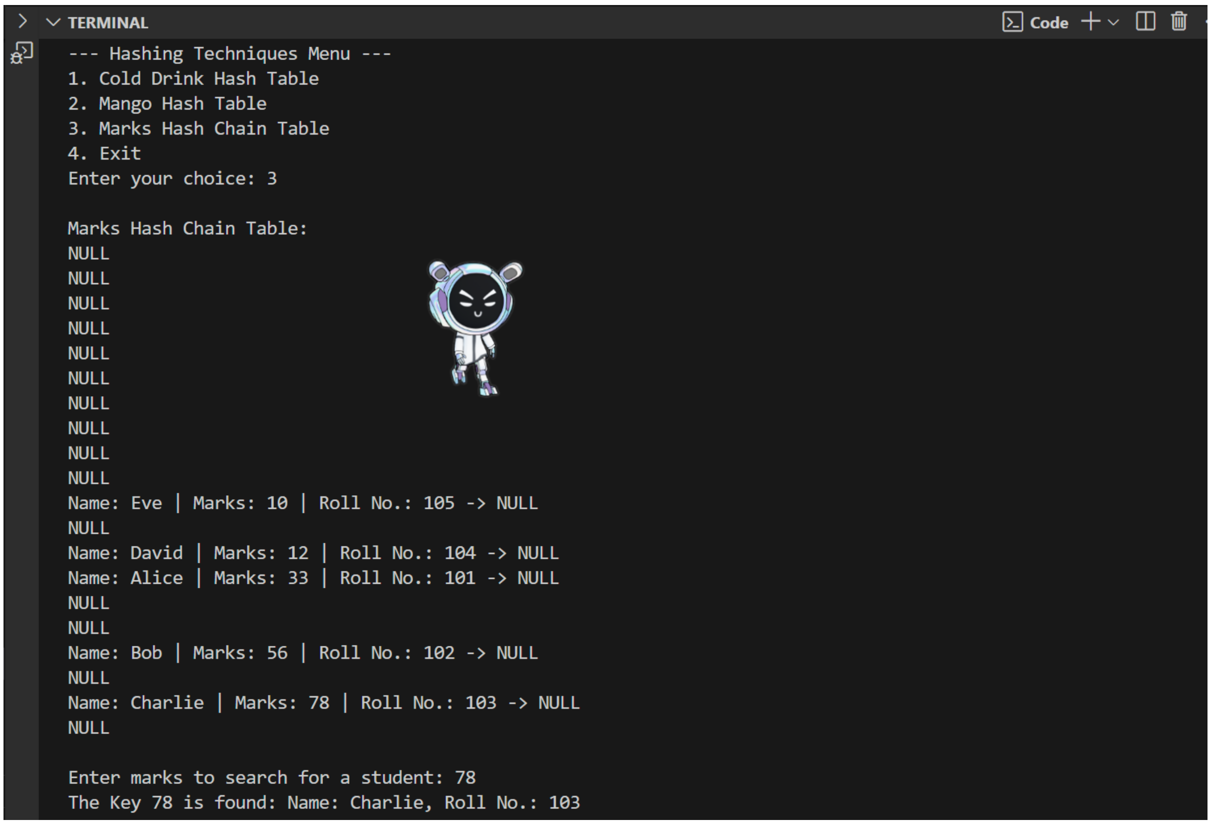
Output:



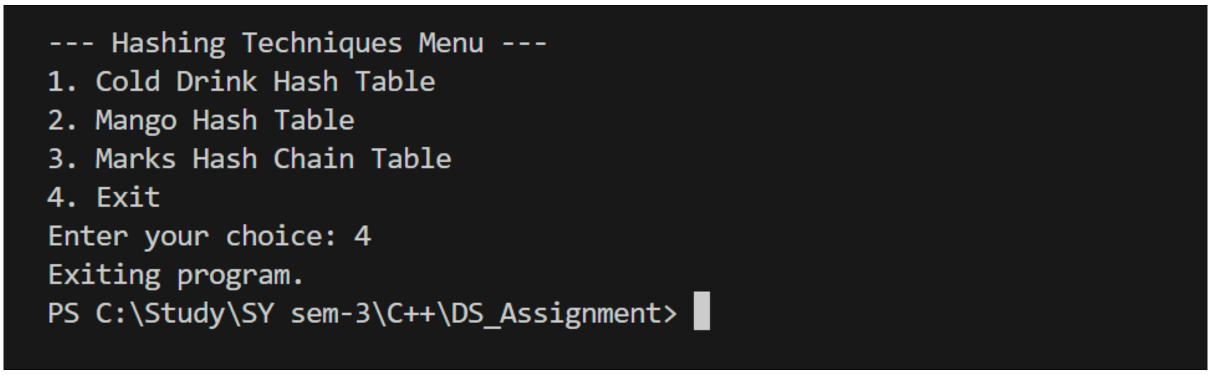
```
PS C:\Study\SY sem-3\C++\DS_Assignment> cd "c:\Study\SY sem-3\C++\DS_Assignment\" ; if ($?) { g++ Practical_1
1.cpp -o Practical_11 } ; if ($?) { .\Practical_11 }
Hash table is full, can't insert more mangoes.

--- Hashing Techniques Menu ---
1. Cold Drink Hash Table
2. Mango Hash Table
3. Marks Hash Chain Table
4. Exit
Enter your choice: 1

Cold Drink Hash Table:
Cold Drink Hash Table:
Index 0: Empty
Index 1: Name: Coke, Price: 1.5, Flavour ID: 101
Index 2: Name: Pepsi, Price: 1.2, Flavour ID: 102
Index 3: Name: Sprite, Price: 1.3, Flavour ID: 103
Index 4: Name: Fanta, Price: 1.4, Flavour ID: 104
Index 5: Empty
Index 6: Empty
Index 7: Empty
Index 8: Empty
Index 9: Empty

Enter flavour ID to search for a cold drink: 102
Cold Drink Found: Pepsi, Price: 1.2, Flavour ID: 102
```



```
--- Hashing Techniques Menu ---
1. Cold Drink Hash Table
2. Mango Hash Table
3. Marks Hash Chain Table
4. Exit
Enter your choice: 2

Mango Varieties Hash Table:
Mango Varieties Hash Table:
Index 0: Category: Category 5, Number of Mangoes: 11
Index 1: Empty
Index 2: Category: Category 11, Number of Mangoes: 32
Index 3: Category: Category 1, Number of Mangoes: 25
Index 4: Category: Category 2, Number of Mangoes: 15
Index 5: Category: Category 4, Number of Mangoes: 5
Index 6: Category: Category 7, Number of Mangoes: 16
Index 7: Category: Category 8, Number of Mangoes: 36
Index 8: Category: Category 6, Number of Mangoes: 19
Index 9: Category: Category 9, Number of Mangoes: 42
Index 10: Category: Category 3, Number of Mangoes: 10

Enter number of mangoes to search for a category: 5
Mango Category Found: Category 4, Number of Mangoes: 5
```



```
--- Hashing Techniques Menu ---
1. Cold Drink Hash Table
2. Mango Hash Table
3. Marks Hash Chain Table
4. Exit
Enter your choice: 3

Marks Hash Chain Table:
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
Name: Eve | Marks: 10 | Roll No.: 105 -> NULL
NULL
Name: David | Marks: 12 | Roll No.: 104 -> NULL
Name: Alice | Marks: 33 | Roll No.: 101 -> NULL
NULL
NULL
Name: Bob | Marks: 56 | Roll No.: 102 -> NULL
NULL
Name: Charlie | Marks: 78 | Roll No.: 103 -> NULL
NULL

Enter marks to search for a student: 78
The Key 78 is found: Name: Charlie, Roll No.: 103
```



```
--- Hashing Techniques Menu ---
1. Cold Drink Hash Table
2. Mango Hash Table
3. Marks Hash Chain Table
4. Exit
Enter your choice: 4
Exiting program.
PS C:\Study\SY sem-3\C++\DS_Assignment>
```