

# Practical 11

NAME: Omkar Londhe


PRN: 202301040027

BATCH: B4

ROLL NO: 59

Code:

```
Practical_11.cpp > Mango
1  #include <iostream>
2  using namespace std;
3
4  class Mango {
5  public:
6      string category;
7      int numberOfMangoes;
8
9      Mango() {
10         category = "";
11         numberOfMangoes = -1;
12     }
13
14     Mango(string cat, int num) {
15         category = cat;
16         numberOfMangoes = num;
17     }
18 };
19
```



```

20 class MangoHashTable {
21 private:
22     Mango* table;
23     int size;
24
25 public:
26     MangoHashTable(int s) {
27         size = s;
28         table = new Mango[size];
29     }
30
31     int hashFunction(int key) {
32         return key % size;
33     }
34
35     void insert(const Mango& mango) {
36         int index = hashFunction(mango.numberOfMangoes);
37         int i = 0;
38         int newIndex = index;
39
40         // Quadratic Probing to handle collisions
41         while (table[newIndex].numberOfMangoes != -1 && i < size) {
42             i++;
43             newIndex = (index + i * i) % size;
44         }
45
46         if (i == size) {
47             cout << "Hash table is full, can't insert more mangoes." << endl;
48             return;
49         }
50
51         table[newIndex] = mango;
52     }
53
54     Mango* search(int key) {
55         int index = hashFunction(key);
56         int i = 0;
57         int newIndex = index;
58
59         while (i < size) {
60             if (table[newIndex].numberOfMangoes == key) {
61                 return &table[newIndex];
62             }
63             i++;
64             newIndex = (index + i * i) % size;
65         }
66         return NULL;
67     }

```

```

69     void display() {
70         cout << "Mango Varieties Hash Table:" << endl;
71         for (int i = 0; i < size; ++i) {
72             if (table[i].numberOfMangoes != -1) {
73                 cout << "Index " << i << ": Category: " << table[i].category
74                     << ", Number of Mangoes: " << table[i].numberOfMangoes << endl;
75             } else {
76                 cout << "Index " << i << ": Empty" << endl;
77             }
78         }
79     }
80 };

```

```

82 class ColdDrink {
83 public:
84     string name;
85     double price;
86     int flavourid;
87
88     ColdDrink() {
89         name = "";
90         price = 0.0;
91         flavourid = -1;
92     }
93
94     ColdDrink(string n, double p, int f) {
95         name = n;
96         price = p;
97         flavourid = f;
98     }
99 };

```

```

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

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

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

```



```

116 void insert(const ColdDrink& drink) {
117     int index = hashFunction(drink.flavourid);
118     int i = 0;
119     int newIndex = index;
120
121     while (table[newIndex].flavourid != -1 && i < size) {
122         i++;
123         newIndex = (index + i * i) % size;
124     }
125
126     if (i == size) {
127         cout << "Hash table is full, can't insert more cold drinks." << endl;
128         return;
129     }
130
131     table[newIndex] = drink;
132 }

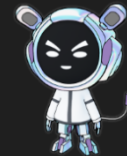
```



```

134 ColdDrink* search(int key) {
135     int index = hashFunction(key);
136     int i = 0;
137     int newIndex = index;
138
139     while (i < size) {
140         if (table[newIndex].flavourid == key) {
141             return &table[newIndex];
142         }
143         i++;
144         newIndex = (index + i * i) % size;
145     }
146     return NULL;
147 }

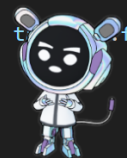
```



```

149 void display() {
150     cout << "Cold Drink Hash Table:" << endl;
151     for (int i = 0; i < size; ++i) {
152         if (table[i].flavourid != -1) {
153             cout << "Index " << i << ": Name: " << table[i].name
154                 << ", Price: " << table[i].price << ", Flavour ID: " << table[i].flavourid << endl;
155         } else {
156             cout << "Index " << i << ": Empty" << endl;
157         }
158     }
159 }
160 };
161

```



```

162 class node {
163 public:
164     string name;
165     int marks;
166     int rollno;
167     node* next;
168     node(string s, int m, int r) {
169         name = s;
170         marks = m;
171         rollno = r;
172         next = NULL;
173     }
174 };
175
176 class hashchain {
177 public:
178     int size3 = 20;
179     node** htable;
180
181     hashchain() {
182         htable = new node*[size3];
183         for (int i = 0; i < size3; i++) {
184             htable[i] = nullptr;
185         }
186     }

```

```

188 void insert(node* temp) {
189     node* p;
190     int h = temp->marks % size3;
191     if (htable[h] == NULL) {
192         htable[h] = temp;
193     } else {
194         p = htable[h];
195         while (p->next != NULL) {
196             p = p->next;
197         }
198         p->next = temp;
199     }
200 }

```

```

202 void search(int key) {
203     int h = key % size3;
204     node* p = htable[h];
205
206     if (p == nullptr) {
207         cout << "The key " << key << " is not present" << endl;
208     } else {
209         bool found = false;
210         while (p != NULL) {
211             if (p->marks == key) {
212                 cout << "The Key " << key << " is found: "
213                     << "Name: " << p->name << ", Roll No.: " << p->rollno << endl;
214                 found = true;
215                 break;
216             }
217             p = p->next;
218         }
219         if (!found) {
220             cout << "The key " << key << " is not present" << endl;
221         }
222     }
223 }

```

```

225     void display() {
226         for (int i = 0; i < size3; i++) {
227             node* p = htable[i];
228             if (p == NULL) {
229                 cout << "NULL" << endl;
230             } else {
231                 while (p != nullptr) {
232                     cout << "Name: " << p->name << " | Marks: " << p->marks << " | Roll No.: ";
233                     p = p->next;
234                 }
235                 cout << "NULL" << endl;
236             }
237         }
238     }
239 };

```

```

241 int main() {
242     int choice;
243     int searchKey;
244
245     int size = 11;
246     MangoHashTable mangoTable(size);
247     ColdDrinkHashTable colddrinkTable(10);
248     hashchain table;
249
250     Mango mangoes[] = {
251         Mango("Category 1", 25),
252         Mango("Category 2", 15),
253         Mango("Category 3", 10),
254         Mango("Category 4", 5),
255         Mango("Category 5", 11),
256         Mango("Category 6", 19),
257         Mango("Category 7", 16),
258         Mango("Category 8", 36),
259         Mango("Category 9", 42),
260         Mango("Category 10", 28),
261         Mango("Category 11", 32)
262     };
263

```

```

264     for (int i = 0; i < size; ++i) {
265         mangoTable.insert(mangoes[i]);
266     }
267
268     colddrinkTable.insert(ColdDrink("Coke", 1.5, 101));
269     colddrinkTable.insert(ColdDrink("Pepsi", 1.2, 102));
270     colddrinkTable.insert(ColdDrink("Sprite", 1.3, 103));
271     colddrinkTable.insert(ColdDrink("Fanta", 1.4, 104));
272
273     table.insert(new node("Alice", 33, 101));
274     table.insert(new node("Bob", 56, 102));
275     table.insert(new node("Charlie", 78, 103));
276     table.insert(new node("David", 12, 104));
277     table.insert(new node("Eve", 10, 105));
278

```

```

279     do {
280         cout << "\n--- Hashing Techniques Menu ---\n";
281         cout << "1. Cold Drink Hash Table\n";
282         cout << "2. Mango Hash Table\n";
283         cout << "3. Marks Hash Chain Table\n";
284         cout << "4. Exit\n";
285         cout << "Enter your choice: ";
286         cin >> choice;
287
288         switch (choice) {
289             case 1: {
290                 cout << "\nCold Drink Hash Table:\n";
291                 colddrinkTable.display();
292                 cout << "\nEnter flavour ID to search for a cold drink: ";
293                 cin >> searchKey;
294                 ColdDrink* coldDrinkResult = colddrinkTable.search(searchKey);
295                 if (coldDrinkResult) {
296                     cout << "Cold Drink Found: " << coldDrinkResult->name
297                         << ", Price: " << coldDrinkResult->price
298                         << ", Flavour ID: " << coldDrinkResult->flavourid << endl;
299                 } else {
300                     cout << "Cold Drink not found!" << endl;
301                 }
302                 break;
303             }

```

```

304             case 2: {
305                 cout << "\nMango Varieties Hash Table:\n";
306                 mangoTable.display();
307                 cout << "\nEnter number of mangoes to search for a category: ";
308                 cin >> searchKey;
309                 Mango* mangoResult = mangoTable.search(searchKey);
310                 if (mangoResult) {
311                     cout << "Mango Category Found: " << mangoResult->category
312                         << ", Number of Mangoes: " << mangoResult->numberOfMangoes << endl;
313                 } else {
314                     cout << "Mango Category not found!" << endl;
315                 }
316                 break;
317             }
318             case 3: {
319                 cout << "\nMarks Hash Chain Table:\n";
320                 table.display();
321                 cout << "\nEnter marks to search for a student: ";
322                 cin >> searchKey;
323                 table.search(searchKey);
324                 break;
325             }

```

```

318         case 3: {
319             cout << "\nMarks Hash Chain Table:\n";
320             table.display();
321             cout << "\nEnter marks to search for a student: ";
322             cin >> searchKey;
323             table.search(searchKey);
324             break;
325         }
326         case 4:
327             cout << "Exiting program." << endl;
328             break;
329         default:
330             cout << "Invalid choice. Please enter a number between 1 and 4." << endl;
331     }
332 } while (choice != 4);
333
334 return 0;
335 }

```

## Output:

```

> v TERMINAL
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

```



```
> ▾ TERMINAL [Code] [+] [ ] [X]
--- 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
```



```
> ▾ TERMINAL [Code] [+] [ ] [X]
--- 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
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> █
```

