

# LAB TASK # 11

## MOHAMMAD BASIL ALI KHAN

### 20K-0477

#### Task#01:

- Code

```
Task#01.cpp x
Task#01.cpp > main()
1  #include<iostream>
2  #define size 10
3
4  using namespace std;
5
6  class HashTable{
7  public:
8      int arr[size];
9
10     HashTable()
11     {
12         for(int i=0; i<size; i++)
13         {
14             arr[i] = 0;
15         }
16     }
17
18     int Hash_Func(int val)
19     {
20         return val % size;
21     }
22
23     void Insertdata(int val)
24     {
25         int index = Hash_Func(val);
26         arr[index] = val;
27     }
28
29     int Display()
30     {
31         for(int i=0; i<size; i++)
32         {
33             if(arr[i]==0)
```

```
Task#01.cpp x
Task#01.cpp > main()
27     }
28
29     int Display()
30     {
31         for(int i=0; i<size; i++)
32         {
33             if(arr[i]==0)
34             {
35                 cout << i << " : " << "NULL" << endl;
36             }
37             else
38             {
39                 cout << i << " : " << arr[i]<< endl;
40             }
41         }
42     }
43 };
44
45
46 int main()
47 {
48     HashTable obj;
49     obj.Insertdata(3);
50     obj.Insertdata(2);
51     obj.Insertdata(9);
52     obj.Insertdata(6);
53     obj.Insertdata(11);
54     obj.Insertdata(13);
55     obj.Insertdata(7);
56     obj.Insertdata(12);
57     obj.Display();
58 }
```

- **Output**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\3rd Semester\DS Lab\Lab Task 11> cd "e:\FAST\3rd Semester\DS Lab\Lab Task 11\" ; if ($?) { g++ Task#01.cpp -o Task#01 } ; if ($?) { .\Task#01 }
0 : NULL
1 : 11
2 : 12
3 : 13
4 : NULL
5 : NULL
6 : 6
7 : 7
8 : NULL
9 : 9
PS E:\FAST\3rd Semester\DS Lab\Lab Task 11> |
```

## Task#02:

```
main.cpp
8
9 #include<iostream>
10 #define size 10
11
12 using namespace std;
13
14 class HashTable{
15 public:
16     int arr[size];
17
18     HashTable()
19     {
20         for(int i=0; i<size; i++)
21         {
22             arr[i] = 0;
23         }
24     }
25
26     void Insertdata(int val)
27     {
28         int index = multiplication_hash(val);
29         arr[index] = val;
30     }
31
32     bool Search(int val)
33     {
34         int index = multiplication_hash(val);
35         for(int i=0; i<size; i++)
36         {
37             if(val == arr[i])
38             {
39                 return true;
40             }
41         }
42         return false;
43     }
44
45     void Delete(int val)
46     {
47         int index = multiplication_hash(val);
48         if(arr[index] == val)
49         {
50             arr[index] = 0;
51             cout << "Value deleted." << endl;
52         }
53     }
54 }
```

```

main.cpp
52
53     void Delete(int val)
54     {
55         int index = multiplication_hash(val);
56         if(arr[index] == val)
57         {
58             arr[index] = 0;
59             cout << "Value deleted." << endl;
60             return;
61         }
62         cout << "Value doesnt exist." << endl;
63     }
64
65     void Display()
66     {
67         for(int i=0; i<size; i++)
68         {
69             if(arr[i]==0)
70             {
71                 cout << i << " -> " << "NULL" << endl;
72             }
73             else
74             {
75                 cout << i << " -> " << arr[i] << endl;
76             }
77         }
78     }
79 };
80
81 int main()
82 {
83     HashTable obj;
84     int n;
85     cout << "Inserting Elements: " << endl;
86     obj.Insertdata(3);
87     obj.Insertdata(4);
88     obj.Insertdata(6);
89     obj.Insertdata(7);
90     obj.Insertdata(10);
91     obj.Insertdata(2);
92     obj.Display();
93 }

```

```

main.cpp
72     }
73     else
74     {
75         cout << i << " -> " << arr[i] << endl;
76     }
77 }
78 };
79
80
81 int main()
82 {
83     HashTable obj;
84     int n;
85     cout << "Inserting Elements: " << endl;
86     obj.Insertdata(3);
87     obj.Insertdata(4);
88     obj.Insertdata(6);
89     obj.Insertdata(7);
90     obj.Insertdata(10);
91     obj.Insertdata(2);
92     obj.Display();
93     cout << endl;
94     cout << "Searching elements: " << endl;
95     cout << "Enter value to search: " << endl;
96     cin >> n;
97     bool ans = obj.Search(n);
98     if(ans==true)
99     {
100         cout << "Value found." << endl;
101     }
102     else
103     {
104         cout << "Value not found. " << endl;
105     }
106     cout << endl;
107     cout << "Deleting Value." << endl;
108     cout << "Enter value to delete: " << endl;
109     cin >> n;
110     obj.Delete(n);
111     obj.Display();
112 }

```

- **Multiplication\_hash**

```

25
26     int multiplication_hash(int key)
27     {
28         float modulu = key * 0.4;
29         int mod = modulu;
30         float f = modulu - mod;
31         return size * f;
32     }
33

```

```
input
Using multiplication_hash:
Inserting Elements:
0 -> 10
1 -> NULL
2 -> 3
3 -> NULL
4 -> 6
5 -> NULL
6 -> 4
7 -> 7
8 -> 2
9 -> NULL

Searching elements:
Enter value to search:
3
Value found.

Deleting Value.
Enter value to delete:
7
Value deleted.
0 -> 10
1 -> NULL
2 -> 3
3 -> NULL
4 -> 6
5 -> NULL
6 -> 4
7 -> NULL
8 -> 2
9 -> NULL

...Program finished with exit code 0
Press ENTER to exit console.
```

- **mid\_square\_hash**

```
33
34     int mid_square_hash(int key)
35     {
36         int value = key*key;
37         int middle_value = value/2;
38         return middle_value;
39     }
40
```

```
input
Using mid_value_hash:
Inserting Elements:
0 -> NULL
1 -> NULL
2 -> 2
3 -> NULL
4 -> 3
5 -> NULL
6 -> NULL
7 -> NULL
8 -> 4
9 -> NULL

Searching elements:
Enter value to search:
3
Value found.

Deleting Value.
Enter value to delete:
4
Value deleted.
0 -> NULL
1 -> NULL
2 -> 2
3 -> NULL
4 -> 3
5 -> NULL
6 -> NULL
7 -> NULL
8 -> NULL
9 -> NULL
```

- **Folding\_hash**

```
40
41     int Folding_hash(int value)
42     {
43         int index = value + value + value;
44         return index % size;
45     }
46
```

```
Using Folding_hash
Inserting Elements:
0 -> 10
1 -> 7
2 -> 4
3 -> NULL
4 -> NULL
5 -> NULL
6 -> 2
7 -> NULL
8 -> 6
9 -> 3

Searching elements:
Enter value to search:
10
Value found.

Deleting Value.
Enter value to delete:
10
Value deleted.
0 -> NULL
1 -> 7
2 -> 4
3 -> NULL
4 -> NULL
5 -> NULL
6 -> 2
7 -> NULL
8 -> 6
9 -> 3
```

- Radix\_hash

```
53 int radix_hash(int value)
54 {
55     int index = value % 1000;
56     return index;
57 }
58
```

```
Using radix hash
Inserting Elements:
0 -> NULL
1 -> NULL
2 -> 2
3 -> 3
4 -> 4
5 -> NULL
6 -> 6
7 -> 7
8 -> NULL
9 -> NULL

Searching elements:
Enter value to search:
6
Value found.

Deleting Value.
Enter value to delete:
4
Value deleted.
0 -> NULL
1 -> NULL
2 -> 2
3 -> 3
4 -> NULL
5 -> NULL
6 -> 6
7 -> 7
8 -> NULL
9 -> NULL
```

## Task#03:

- Code

```
Task#03.cpp X
Task#03.cpp > main()
1  #include<iostream>
2  #define size 10
3
4  using namespace std;
5
6  class Node{
7  public:
8      int contact;
9      int ID;
10
11      Node()
12      {
13          ID = 0;
14          contact = 0;
15      }
16
17      Node(int val1, int val2)
18      {
19          ID = val1;
20          contact = val2;
21      }
22 };
23
24 class HashTable{
25 public:
26     Node *arr[size];
27
28     HashTable()
29     {
30         for(int i=0; i<size; i++)
31         {
32             arr[i] = NULL;
33         }
34     }
35 }
```

```
Task#03.cpp X
Task#03.cpp > main()
62 void IsEmpty()
63 {
64     for(int i=0; i<size; i++)
65     {
66         if(arr[i]!=NULL)
67         {
68             cout << "Contact Book not empty. " << endl;
69             return ;
70         }
71     }
72     cout << "Contact Book empty. " << endl;
73 }
74
75 int ContactSize()
76 {
77     return size;
78 }
79
80 bool Search(int val)
81 {
82     int index = Hash_Func(val);
83     if(arr[index]->ID==val)
84     {
85         cout << "Entered Record found at location: " << index << endl;
86         cout << "Student ID: " << arr[index]->ID << endl;
87         cout << "Contact Number: " << arr[index]->contact << endl;
88         return true;
89     }
90     else
91     {
92         for(int i=0; i<size; i++)
93         {
94             if(arr[i]->ID==val)
95             {
96                 cout << "Entered Record found at location: " << i << endl;
97                 cout << "Student ID: " << arr[i]->ID << endl;
98                 cout << "Contact Number: " << arr[i]->contact << endl;
99                 return true;
100             }
101         }
102     }
103     return false;
104 }
```

```
Task#03.cpp x
Task#03.cpp > main()
122     }
123     else
124     {
125         for(int i=0; i<size; i++)
126         {
127             int z = (index+i) % size;
128             if(arr[z]->ID==val)
129             {
130                 delete arr[index];
131                 arr[index] = 0;
132                 return;
133             }
134         }
135     }
136 }
137
138
139 int Display()
140 {
141     for(int i=0; i<size; i++)
142     {
143         if(arr[i]==NULL)
144         {
145             cout << i << " -> " << "NULL" << endl;
146         }
147         else
148         {
149             cout << i << " -> " << arr[i]->ID << " " << arr[i]->contact << endl;
150         }
151     }
152 }
153 };
154
```

## • Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\3rd Semester\DS Lab\Lab Task 11> cd "e:\FAST\3rd Semester\DS Lab\Lab Task 11\" ; if ($?) { g++ Task#03.cpp -o Task#03 } ; if ($?) { .\Task#03 }
Before entering record:
Whether book is empty or not: Contact Book empty.

Contact Book Size: 10

Inserting record:
0 -> 200877 7875126
1 -> NULL
2 -> NULL
3 -> NULL
4 -> NULL
5 -> NULL
6 -> NULL
7 -> 200477 113238
8 -> 245477 32478126
9 -> 206748 485126

Whether book is empty or not: Contact Book not empty.

Searching a record:
Entered Record found at location: 7
Student ID: 200477
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
6 -> NULL
7 -> 200477 113238
8 -> 245477 32478126
9 -> 206748 485126

Whether book is empty or not: Contact Book not empty.

Searching a record:
Entered Record found at location: 7
Student ID: 200477
Contact Number: 113238

Deleting record:
Entered Record found at location: 8
Student ID: 245477
Contact Number: 32478126

After deleting record:
0 -> 200877 7875126
1 -> NULL
2 -> NULL
3 -> NULL
4 -> NULL
5 -> NULL
6 -> NULL
7 -> 200477 113238
8 -> NULL
9 -> 206748 485126
PS E:\FAST\3rd Semester\DS Lab\Lab Task 11>
```

## Task#04:

- Code

```
Task#04.cpp X
Task#04.cpp > HashTable > Delete(int)
1 #include<iostream>
2 #define size 7
3
4 using namespace std;
5
6 class Node{
7     public:
8         int contact;
9         int ID;
10        Node *next;
11
12        Node()
13        {
14            ID = 0;
15            contact = 0;
16            next = NULL;
17        }
18
19        Node(int val1, int val2)
20        {
21            ID = val1;
22            contact = val2;
23            next = NULL;
24        }
25    };
26
27 class HashTable{
28     public:
29         Node *chain[size];
30
31         HashTable()
32         {
33             for(int i=0; i<size; i++)
```



```
Task#04.cpp X
Task#04.cpp > HashTable > Delete(int)
30
31     HashTable()
32     {
33         for(int i=0; i<size; i++)
34         {
35             chain[i] = NULL;
36         }
37     }
38
39     int Hash_Func(int val)
40     {
41         return val % size;
42     }
43
44     void Insertdata(int val1, int val2)
45     {
46         Node *n = new Node(val1, val2);
47         int index = Hash_Func(val1);
48         if(chain[index]==NULL)
49         {
50             chain[index] = n;
51         }
52         else
53         {
54             Node *temp = chain[index];
55             while(temp->next!=NULL)
56             {
57                 temp = temp->next;
58             }
59             temp->next = n;
60         }
61     }
62
```

```
Task#04.cpp X
Task#04.cpp > HashTable > Delete(int)
63     int ContactSize()
64     {
65         return size;
66     }
67
68     void IsEmpty()
69     {
70         for(int i=0; i<size; i++)
71         {
72             if(chain[i]!=NULL)
73             {
74                 cout << "Contact Book not empty. " << endl;
75                 return ;
76             }
77         }
78         cout << "Contact Book empty. " << endl;
79     }
80
81     void Search(int val)
82     {
83         int index = val % size;
84         Node *temp = chain[index];
85         while(temp!=NULL)
86         {
87             if(temp->ID==val)
88             {
89                 cout << "Entered Record found at location: " << index << endl;
90                 cout << "Student ID: " << chain[index]->ID << endl;
91                 cout << "Contact Number: " << chain[index]->contact << endl;
92                 return;
93             }
94             temp = temp->next;
95         }
96     }
97
```

```
Task#04.cpp X
Task#04.cpp > HashTable > Delete(int)
93     }
94     temp = temp->next;
95 }
96 cout << "Value not found." << endl;
97 }
98
99 void Delete(int val)
100 {
101     int index = val % size;
102     if(chain[index]!=NULL)
103     {
104         if(chain[index]->ID==val)
105         {
106             cout << "Entered Record found at location: " << index << endl;
107             cout << "Student ID: " << chain[index]->ID << endl;
108             cout << "Contact Number: " << chain[index]->contact << endl;
109             delete chain[index];
110             chain[index] = NULL;
111             cout << "Element deleted." << endl;
112             return;
113         }
114         else
115         {
116             Node *temp = chain[index];
117             while(temp->next!=NULL)
118             {
119                 if(temp->next->ID==val)
120                 {
121                     cout << "Entered Record found at location: " << index << endl;
122                     cout << "Student ID: " << chain[index]->ID << endl;
123                     cout << "Contact Number: " << chain[index]->contact << endl;
124                     Node *del = temp->next;
125                     temp->next = temp->next->next;
```

```
Task#04.cpp X
Task#04.cpp > HashTable > Delete(int)
120     }
121     {
122         cout << "Entered Record found at location: " << index << endl;
123         cout << "Student ID: " << chain[index]->ID << endl;
124         cout << "Contact Number: " << chain[index]->contact << endl;
125         Node *del = temp->next;
126         temp->next = temp->next->next;
127         delete del;
128         cout << "Element deleted." << endl;
129         return;
130     }
131     temp = temp->next;
132 }
133 }
134 cout << "Element doesnt exist." << endl;
135 }
136
137 void Display()
138 {
139     for(int i=0; i<size; i++)
140     {
141         Node *temp = chain[i];
142         cout << i << " : ";
143         while(temp!=NULL)
144         {
145             cout << " | " << temp->ID << " " << temp->contact << " | " << " -> ";
146             temp = temp->next;
147         }
148         cout << "NULL" << endl;
149     }
150 }
151 };
152
```

```
Task#04.cpp x
Task#04.cpp > HashTable > Delete(int)

153
154 int main()
155 {
156     HashTable obj;
157     cout << "Before entering record: " << endl;
158     cout << "Whether book is empty or not: ";
159     obj.IsEmpty();
160     cout << endl;
161     cout << "Contact Book Size: ";
162     cout << obj.ContactSize() << endl;
163     cout << endl;
164     cout << "Inserting record: " << endl;
165     obj.Insertdata(200477, 0335126);
166     obj.Insertdata(245477, 32478126);
167     obj.Insertdata(206748, 485126);
168     obj.Insertdata(200877, 7875126);
169     obj.Insertdata(1, 7875126);
170     obj.Display();
171     cout << endl;
172     cout << "Whether book is empty or not: ";
173     obj.IsEmpty();
174     cout << endl;
175     cout << "Searching a record: " << endl;
176     obj.Search(200477);
177     cout << endl;
178     cout << "Deleting record: " << endl;
179     obj.Delete(206748);
180     cout << endl;
181     cout << "After deleting record: " << endl;
182     obj.Display();
183 }
```

## • Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
04 }
Before entering record:
Whether book is empty or not: Contact Book empty.

Contact Book Size: 7

Inserting record:
0 : NULL
1 : | 245477 32478126 | -> | 1 7875126 | -> NULL
2 : NULL
3 : | 206748 485126 | -> NULL
4 : | 200477 113238 | -> NULL
5 : | 200877 7875126 | -> NULL
6 : NULL

Whether book is empty or not: Contact Book not empty.

Searching a record:
Entered Record found at location: 4
Student ID: 200477
Contact Number: 113238

Deleting record:
Entered Record found at location: 3
Student ID: 206748
Contact Number: 485126
Element deleted.

After deleting record:
0 : NULL
1 : | 245477 32478126 | -> | 1 7875126 | -> NULL
2 : NULL
3 : NULL
4 : | 200477 113238 | -> NULL
5 : | 200877 7875126 | -> NULL
6 : NULL
PS E:\FAST\3rd Semester\DS Lab\Lab Task 11>
```