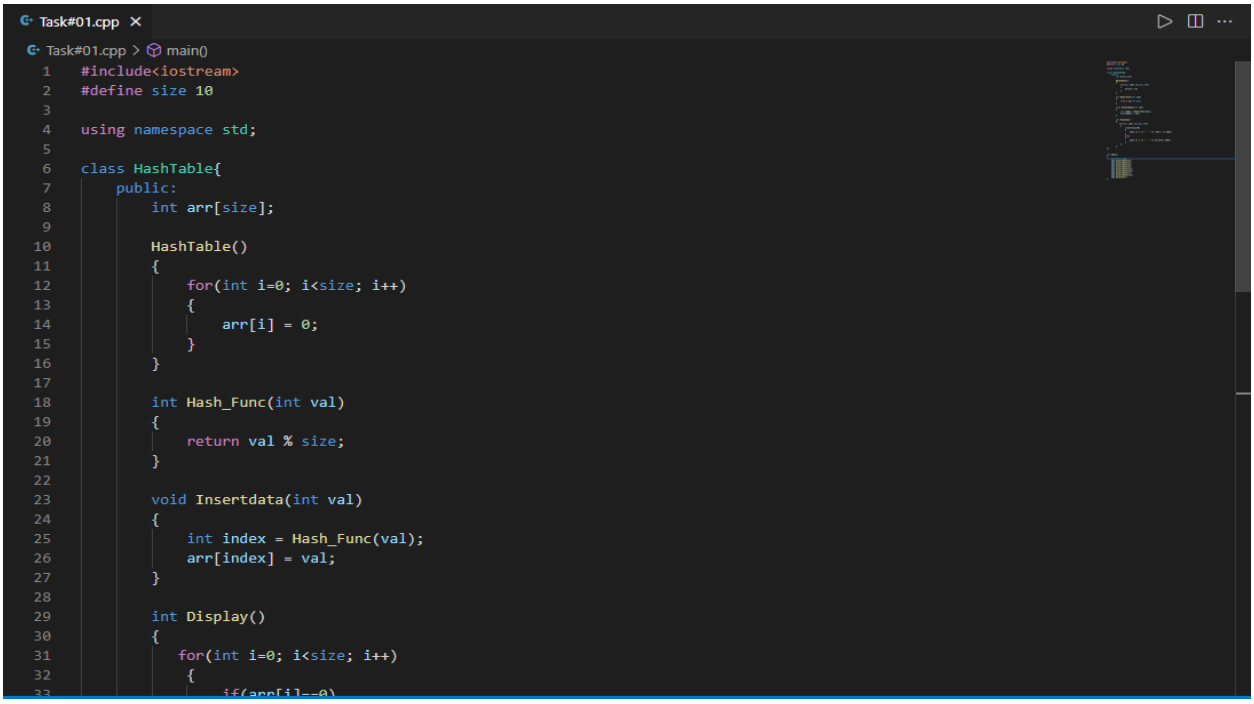
LAB TASK # 11

MOHAMMAD BASIL ALI KHAN

20K-0477

Task#01:

- **Code**



```cpp
#include<iostream>
#define size 10

using namespace std;

class HashTable{
    public:
        int arr[size];

        HashTable()
        {
            for(int i=0; i<size; i++)
            {
                arr[i] = 0;
            }
        }

        int Hash_Func(int val)
        {
            return val % size;
        }

        void Insertdata(int val)
        {
            int index = Hash_Func(val);
            arr[index] = val;
        }

        int Display()
        {
           for(int i=0; i<size; i++)
            {
                if(arr[i]==0)
                {
                    cout << i << " : " << "NULL" << endl;
                }
                else
                {
                    cout << i << " : " << arr[i]<< endl;
                }
            }
        }
};


int main()
{
    HashTable obj;
    obj.Insertdata(3);
    obj.Insertdata(2);
    obj.Insertdata(9);
    obj.Insertdata(6);
    obj.Insertdata(11);
    obj.Insertdata(13);
    obj.Insertdata(7);
    obj.Insertdata(12);
    obj.Display();
}
```

- **Output**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                   Code

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

```cpp
#include<iostream>
#define size 10

using namespace std;

class HashTable{
    public:
        int arr[size];

        HashTable()
        {
            for(int i=0; i<size; i++)
            {
                arr[i] = 0;
            }
        }

        void Insertdata(int val)
        {
            int index = multiplication_hash(val);
            arr[index] = val;
        }

        bool Search(int val)
        {
            int index = multiplication_hash(val);
            for(int i=0; i<size; i++)
            {
                if(val == arr[i])
                {
                    return true;
                }
            }
            return false;
        }

        void Delete(int val)
        {
            int index = multiplication_hash(val);
            if(arr[index] == val)
            {
                arr[index] = 0;
                cout << "Value deleted." << endl;
```

main.cpp

```
    void Delete(int val)
    {
        int index = multiplication_hash(val);
        if(arr[index] == val)
        {
            arr[index] = 0;
            cout << "Value deleted." << endl;
            return;
        }
        cout << "Value doesnt exist." << endl;
    }

    void Display()
    {
        for(int i=0; i<size; i++)
        {
            if(arr[i]==0)
            {
                cout << i << " -> " << "NULL" << endl;
            }
            else
            {
                cout << i << " -> " << arr[i] << endl;
            }
        }
    }
};

int main()
{
    HashTable obj;
    int n;
    cout << "Inserting Elements: " << endl;
    obj.Insertdata(3);
    obj.Insertdata(4);
    obj.Insertdata(6);
    obj.Insertdata(7);
    obj.Insertdata(10);
    obj.Insertdata(2);
    obj.Display();
```

main.cpp

```
                }
                else
                {
                    cout << i << " -> " << arr[i] << endl;
                }
            }
        }
};

int main()
{
    HashTable obj;
    int n;
    cout << "Inserting Elements: " << endl;
    obj.Insertdata(3);
    obj.Insertdata(4);
    obj.Insertdata(6);
    obj.Insertdata(7);
    obj.Insertdata(10);
    obj.Insertdata(2);
    obj.Display();
    cout << endl;
    cout << "Searching elements: " << endl;
    cout << "Enter value to search: " << endl;
    cin >> n;
    bool ans = obj.Search(n);
    if(ans==true)
    {
        cout << "Value found." << endl;
    }
    else
    {
        cout << "Value not found. " << endl;
    }
    cout << endl;
    cout << "Deleting Value." << endl;
    cout << "Enter value to delete: " << endl;
    cin >> n;
    obj.Delete(n);
    obj.Display();
}
```

- **Multiplication_hash**

```
    int multiplication_hash(int key)
    {
        float modulu = key * 0.4;
        int mod = modulu;
        float f = modulu - mod;
        return size * f;
    }
```

```
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
```

```
input
    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**

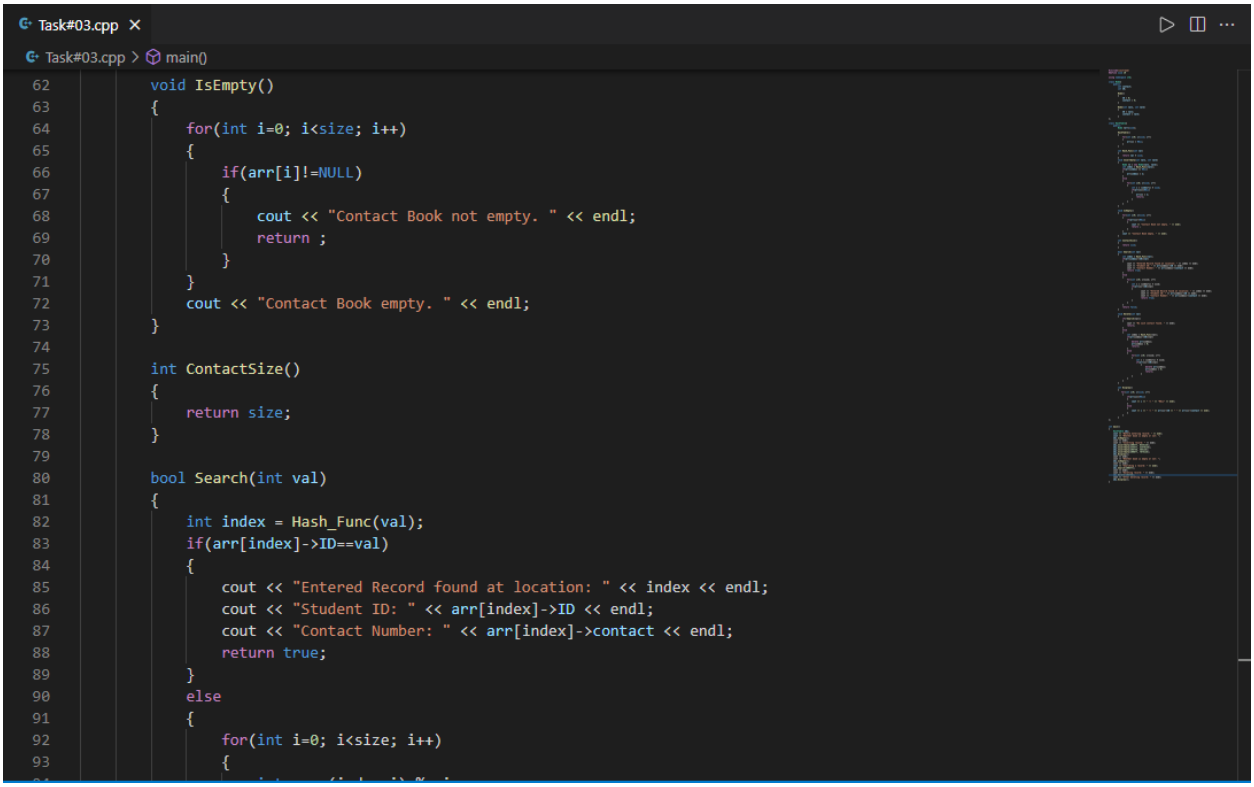
```
53      int radix_hash(int value)
54      {
55          int index = value % 1000;
56          return index;
57  }
58
```

```
input
    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**

```cpp
#include<iostream>
#define size 10

using namespace std;

class Node{
    public:
        int contact;
        int ID;

        Node()
        {
            ID = 0;
            contact = 0;
        }

        Node(int val1, int val2)
        {
            ID = val1;
            contact = val2;
        }
};

class HashTable{
    public:
        Node *arr[size];

        HashTable()
        {
            for(int i=0; i<size; i++)
            {
                arr[i] = NULL;
```



```cpp
        void IsEmpty()
        {
            for(int i=0; i<size; i++)
            {
                if(arr[i]!=NULL)
                {
                    cout << "Contact Book not empty. " << endl;
                    return ;
                }
            }
            cout << "Contact Book empty. " << endl;
        }

        int ContactSize()
        {
            return size;
        }

        bool Search(int val)
        {
            int index = Hash_Func(val);
            if(arr[index]->ID==val)
            {
                cout << "Entered Record found at location: " << index << endl;
                cout << "Student ID: " << arr[index]->ID << endl;
                cout << "Contact Number: " << arr[index]->contact << endl;
                return true;
            }
            else
            {
                for(int i=0; i<size; i++)
                {
```

Task#03.cpp

```cpp
            }
            else
            {
                for(int i=0; i<size; i++)
                {
                    int z = (index+i) % size;
                    if(arr[z]->ID==val)
                        {
                            delete arr[index];
                            arr[index] = 0;
                            return;
                        }
                }
            }
        }

        int Display()
        {
           for(int i=0; i<size; i++)
            {
                if(arr[i]==NULL)
                {
                    cout << i << " -> " << "NULL" << endl;
                }
                else
                {
                    cout << i << " -> " << arr[i]->ID << " " << arr[i]->contact << endl;
                }
            }
        }
};
```

- **Output**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL                                                    Code

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                                Code
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

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

```cpp
#include<iostream>
#define size 7

using namespace std;

class Node{
    public:
        int contact;
        int ID;
        Node *next;

        Node()
        {
            ID = 0;
            contact = 0;
            next = NULL;
        }

        Node(int val1, int val2)
        {
            ID = val1;
            contact = val2;
            next = NULL;
        }
};

class HashTable{
    public:
        Node *chain[size];

        HashTable()
        {
            for(int i=0; i<size; i++)
```

Task#04.cpp

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

```cpp
    HashTable()
    {
        for(int i=0; i<size; i++)
        {
            chain[i] = NULL;
        }
    }

    int Hash_Func(int val)
    {
        return val % size;
    }

    void Insertdata(int val1, int val2)
    {
        Node *n = new Node(val1, val2);
        int index = Hash_Func(val1);
        if(chain[index]==NULL)
        {
            chain[index] = n;
        }
        else
        {
            Node *temp = chain[index];
            while(temp->next!=NULL)
            {
                temp = temp->next;
            }
            temp->next = n;
        }
    }

```

Task#04.cpp

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

```cpp
    int ContactSize()
    {
        return size;
    }

    void IsEmpty()
    {
        for(int i=0; i<size; i++)
        {
            if(chain[i]!=NULL)
            {
                cout << "Contact Book not empty. " << endl;
                return ;
            }
        }
        cout << "Contact Book empty. " << endl;
    }

    void Search(int val)
    {
        int index = val % size;
        Node *temp = chain[index];
        while(temp!=NULL)
        {
            if(temp->ID==val)
            {
                cout << "Entered Record found at location: " << index << endl;
                cout << "Student ID: " << chain[index]->ID << endl;
                cout << "Contact Number: " << chain[index]->contact << endl;
                return;
            }
            temp = temp->next;
```

Task#04.cpp

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

```
            }
            temp = temp->next;
        }
        cout << "Value not found." << endl;
    }

    void Delete(int val)
    {
        int index = val % size;
        if(chain[index]!=NULL)
        {
            if(chain[index]->ID==val)
            {
                cout << "Entered Record found at location: " << index << endl;
                cout << "Student ID: " << chain[index]->ID << endl;
                cout << "Contact Number: " << chain[index]->contact << endl;
                delete chain[index];
                chain[index] = NULL;
                cout << "Element deleted." << endl;
                return;
            }
            else
            {
                Node *temp = chain[index];
                while(temp->next!=NULL)
                {
                    if(temp->next->ID==val)
                    {
                        cout << "Entered Record found at location: " << index << endl;
                        cout << "Student ID: " << chain[index]->ID << endl;
                        cout << "Contact Number: " << chain[index]->contact << endl;
                        Node *del = temp->next;
                        temp->next = temp->next->next;
```

Task#04.cpp

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

```
                    {
                        cout << "Entered Record found at location: " << index << endl;
                        cout << "Student ID: " << chain[index]->ID << endl;
                        cout << "Contact Number: " << chain[index]->contact << endl;
                        Node *del = temp->next;
                        temp->next = temp->next->next;
                        delete del;
                        cout << "Element deleted." << endl;
                        return;
                    }
                    temp = temp->next;
                }
            }
        }
        cout << "Element doesnt exist." << endl;
    }

    void Display()
    {
        for(int i=0; i<size; i++)
        {
            Node *temp = chain[i];
            cout << i << " : ";
            while(temp!=NULL)
            {
                cout << " | "  << temp->ID << " " << temp->contact << " | " << " -> ";
                temp = temp->next;
            }
            cout << "NULL" << endl;
        }
    }
};
```

Task#04.cpp

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

```cpp

int main()
{
    HashTable obj;
    cout << "Before entering record: " << endl;
    cout << "Whether book is empty or not: ";
    obj.IsEmpty();
    cout << endl;
    cout << "Contact Book Size: ";
    cout << obj.ContactSize() << endl;
    cout << endl;
    cout << "Inserting record: " << endl;
    obj.Insertdata(200477, 0335126);
    obj.Insertdata(245477, 32478126);
    obj.Insertdata(206748, 485126);
    obj.Insertdata(200877, 7875126);
    obj.Insertdata(1, 7875126);
    obj.Display();
    cout << endl;
    cout << "Whether book is empty or not: ";
    obj.IsEmpty();
    cout << endl;
    cout << "Searching a record: " << endl;
    obj.Search(200477);
    cout << endl;
    cout << "Deleting record: " << endl;
    obj.Delete(206748);
    cout << endl;
    cout << "After deleting record: " << endl;
    obj.Display();
}
```

- **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>
```