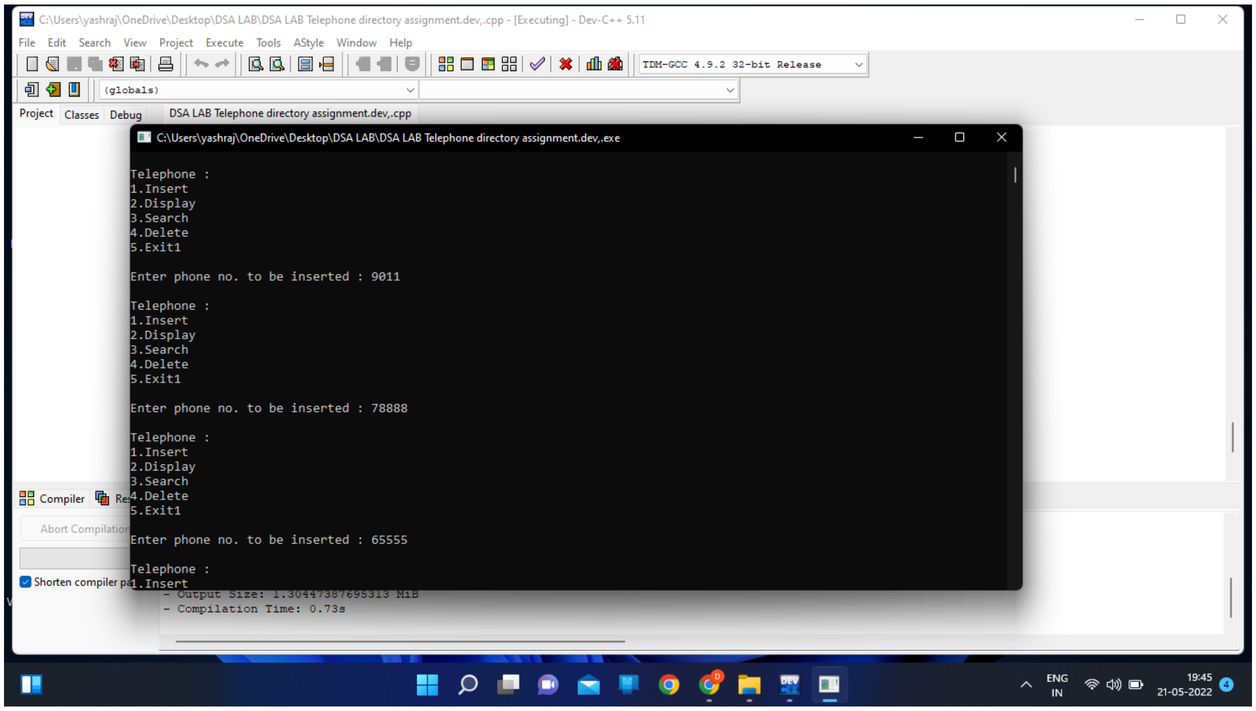
```
#include<iostream>
#include<string.h>
using namespace std;

 struct node
{
 int value;
            node* next;
}*HashTable[10];

class hashing
{
public:
hashing()
{
for(int i=0 ; i<10 ; i++)
{
                        HashTable[i]=NULL;
            }
 }

int HashFunction(int value)
{
  return (value%10);
 }
 node* create_node(int x)
{
             node* temp=new node;
            temp->next=NULL;
            temp->value=x;
            return temp;
 }

 void display()
{
             for(int i=0 ; i< 10; i++)
{
                        node * temp=new node;
                        temp=HashTable[i];
                        cout<<"a["<<i<<"] : ";
                        while(temp !=NULL)
{
                                    cout<<" ->"<<temp->value;
```

```
                                temp=temp->next;
                      }
                     cout<<"\n";
}
 }

int searchElement(int value)
{
            bool flag = false;
             int hash_val = HashFunction(value);
             node* entry = HashTable[hash_val];
             cout<<"\nElement found at : ";
             while (entry != NULL)
            {
                     if (entry->value==value)
                      {
                                cout<<hash_val<<" : "<<entry->value<<endl;
                                flag = true;
                      }
                     entry = entry->next;
             }
             if (!flag)
             return -1;
 }

 void deleteElement(int value)
{
            int hash_val = HashFunction(value);
             node* entry = HashTable[hash_val];
             if (entry == NULL )
             {
                      cout<<"No Element found ";
                       return;
             }

             if(entry->value==value)
{
                     HashTable[hash_val]=entry->next;
                     return;
             }
             while ((entry->next)->value != value)
           {
                entry = entry->next;
             }
            entry->next=(entry->next)->next;
 }

 void insertElement(int value)
{
            int hash_val = HashFunction(value);
            // node* prev = NULL;
             //node* entry = HashTable[hash_val];
             node* temp=new node;
             node* head=new node;
             head = create_node(value);
             temp=HashTable[hash_val];
             if (temp == NULL)
                                   {
                          HashTable[hash_val] =head;
                           }
             else
{
                      while (temp->next != NULL)
                     {
                       temp = temp->next;
                        }
                                temp->next =head;
             }
 }
};
int main()
{
           int ch;
    int data,search,del;
           hashing h;

           do
{

cout<<"\nTelephone : \n1.Insert \n2.Display \n3.Search\n4.Delete \n5.Exit";
                      cin>>ch;
                      switch(ch)
{
                                 case 1:cout<<"\nEnter phone no. to be inserted : ";
                                            cin>>data;
h.insertElement(data);
                                             break;
                                 case 2:h.display();
```

```
                                            break;
                                case 3:cout<<"\nEnter the no to be searched : ";
                                            cin>>search;

                                             if (h.searchElement(search) == -1)
                                            {
                                                                        cout<<"No element found at key ";
                                                                         continue;
                                            }
                                            break;
                                case 4:cout<<"\nEnter the phno. to be deleted : ";
                                             cin>>del;
                                             h.deleteElement(del);
                                              cout<<"Phno. Deleted"<<endl;
                                            break;
                                            }
                        }while(ch!=5);
                        return 0;

}
```

```
#include<iostream>
#include<string.h>
using namespace std;

 struct node
{
 int value;
            node* next;
}*HashTable[10];

class hashing
{
public:
hashing()
{
for(int i=0 ; i<10 ; i++)
{
                        HashTable[i]=NULL;
            }
 }

int HashFunction(int value)
{
  return (value%10);
 }
 node* create_node(int x)
{
             node* temp=new node;
            temp->next=NULL;
            temp->value=x;
            return temp;
 }

 void display()
{
             for(int i=0 ; i< 10; i++)
{
                        node * temp=new node;
                        temp=HashTable[i];
                        cout<<"a["<<i<<"] : ";
                        while(temp !=NULL)
{
                                    cout<<" ->"<<temp->value;
                                    temp=temp->next;
                         }
                        cout<<"\n";
}
 }

int searchElement(int value)
{
             bool flag = false;
              int hash_val = HashFunction(value);
              node* entry = HashTable[hash_val];
              cout<<"\nElement found at : ";
              while (entry != NULL)
             {
                        if (entry->value==value)
                         {
                                    cout<<hash_val<<" : "<<entry->value<<endl;
                                    flag = true;
                         }
                        entry = entry->next;
              }
              if (!flag)
              return -1;
 }

 void deleteElement(int value)
{
             int hash_val = HashFunction(value);
              node* entry = HashTable[hash_val];
              if (entry == NULL )
              {
                         cout<<"No Element found ";
                          return;
              }

              if(entry->value==value)
{
                        HashTable[hash_val]=entry->next;
                        return;
              }
              while ((entry->next)->value != value)
            {
                 entry = entry->next;
              }
              entry->next=(entry->next)->next;
 }

 void insertElement(int value)
{
             int hash_val = HashFunction(value);
             // node* prev = NULL;
              //node* entry = HashTable[hash_val];
              node* temp=new node;
              node* head=new node;
              head = create_node(value);
              temp=HashTable[hash_val];
              if (temp == NULL)
                                  {
                           HashTable[hash_val] =head;
                            }
              else
{
                         while (temp->next != NULL)
                        {
                          temp = temp->next;
                           }
                                    temp->next =head;
              }
 }
};
int main()
{
             int ch;
      int data,search,del;
             hashing h;

             do
{

cout<<"\nTelephone : \n1.Insert \n2.Display \n3.Search\n4.Delete \n5.Exit";
                         cin>>ch;
                         switch(ch)
{
                               case 1:cout<<"\nEnter phone no. to be inserted : ";
                                         cin>>data;
h.insertElement(data);
                                          break;
                               case 2:h.display();
                                         break;
                               case 3:cout<<"\nEnter the no to be searched : ";
                                         cin>>search;

                                          if (h.searchElement(search) == -1)
                                         {
                                                             cout<<"No element found at key ";
                                                              continue;
                                         }
                                         break;
                               case 4:cout<<"\nEnter the phno. to be deleted : ";
                                          cin>>del;
                                          h.deleteElement(del);
                                           cout<<"Phno. Deleted"<<endl;
                                         break;
                                         }
                         }while(ch!=5);
                         return 0;

}
```



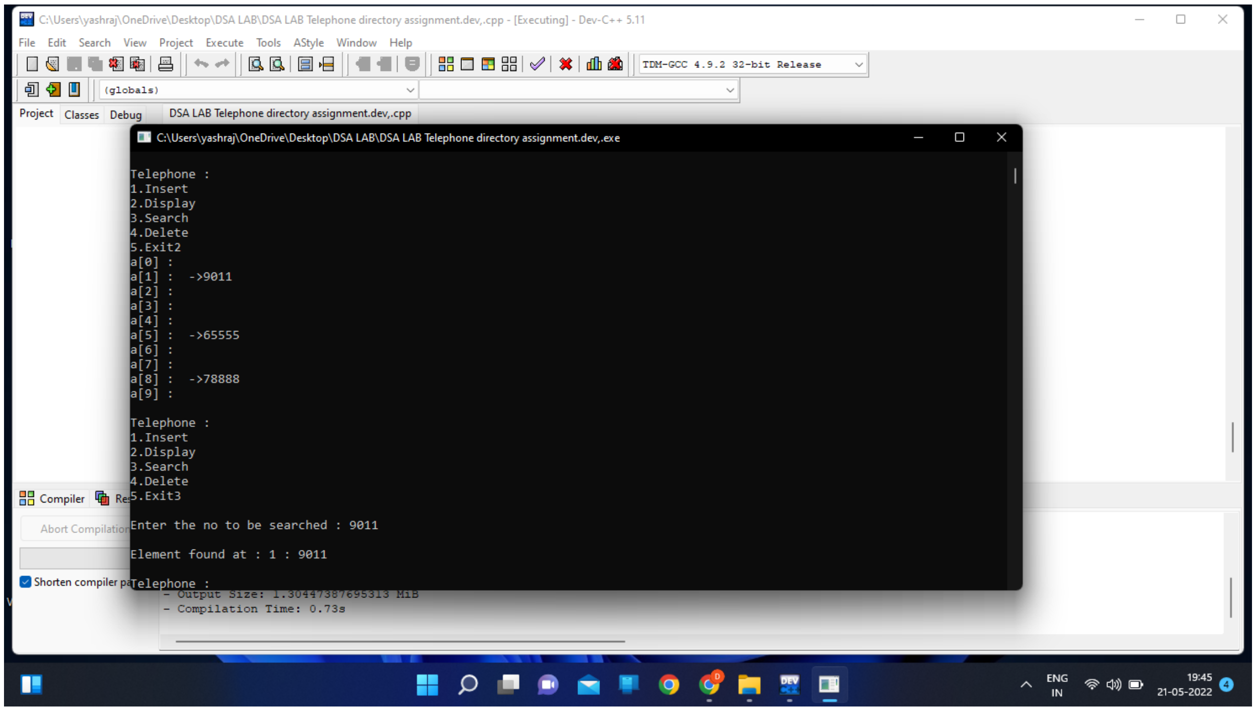
C:\Users\yashraj\OneDrive\Desktop\DSA LAB\DSA LAB Telephone directory assignment.dev,.cpp - [Executing] - Dev-C++ 5.11

C:\Users\yashraj\OneDrive\Desktop\DSA LAB\DSA LAB Telephone directory assignment.dev,.exe

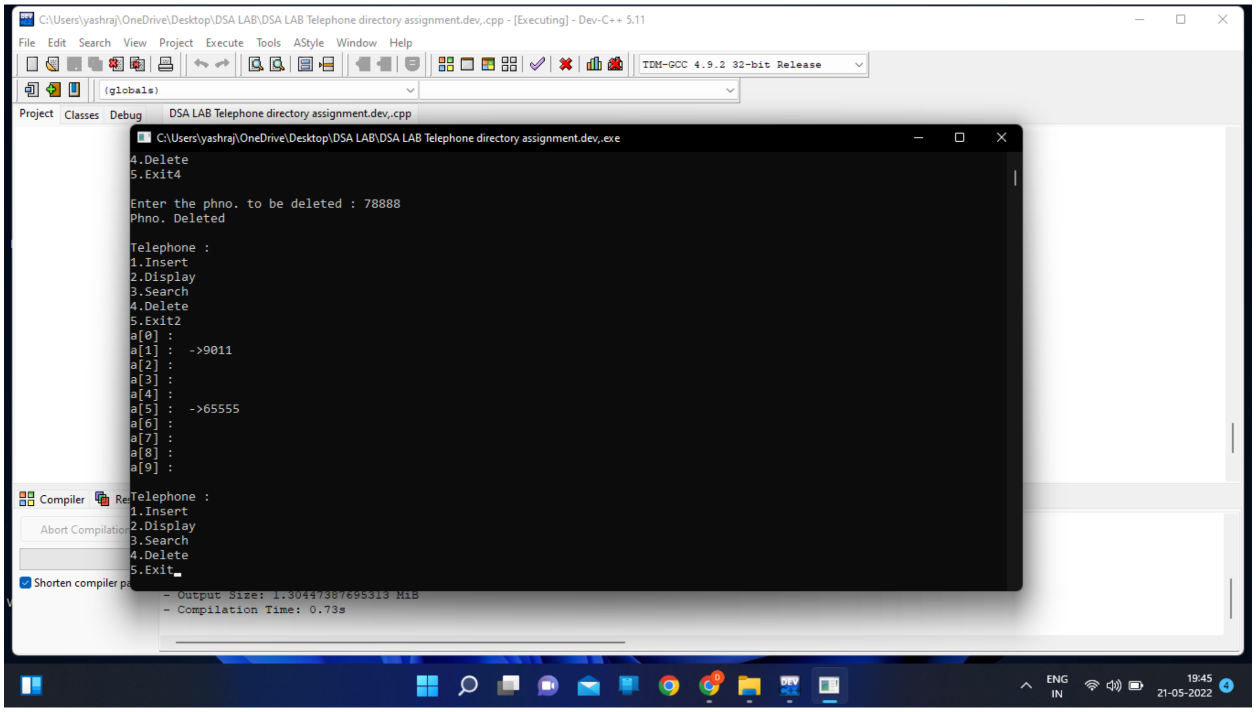
```
Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit1

Enter phone no. to be inserted : 9011

Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit1

Enter phone no. to be inserted : 78888

Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit1

Enter phone no. to be inserted : 65555

Telephone :
1.Insert
```

```
- Output Size: 1.30447387695313 MiB
- Compilation Time: 0.73s
```

File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
DSA LAB Telephone directory assignment.dev,.cpp
Compiler
Abort Compilation
Shorten compiler pa
ENG IN
19:45
21-05-2022



```
Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit2
a[0] :
a[1] :  ->9011
a[2] :
a[3] :
a[4] :
a[5] :  ->65555
a[6] :
a[7] :
a[8] :  ->78888
a[9] :

Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit3

Enter the no to be searched : 9011

Element found at : 1 : 9011

Telephone :
```



```
4.Delete
5.Exit4

Enter the phno. to be deleted : 78888
Phno. Deleted

Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit2
a[0] :
a[1] :  ->9011
a[2] :
a[3] :
a[4] :
a[5] :  ->65555
a[6] :
a[7] :
a[8] :
a[9] :

Telephone :
1.Insert
2.Display
3.Search
4.Delete
5.Exit_
```