10. Shabbir Ezzy

Title: Implement all the functions of a dictionary (ADT) using hashing and handle collisions using chaining with/without replacement. Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable, Keys must be unique. Standard Operations: Insert(key, value), Find (key), Delete(key).

```
def display(ff):
        output = "{"
        for i in range(len(ff)):
            output += str(ff[i])
            if i < len(ff) - 1:
                output += ", "
        output += "}"
        print(output)

class SET:
    def __init__(self):
        self.a=[]
        self.b=[]



        self.sizeA=int(input("enter the size of Set A "))
        self.sizeB = int(input("enter the size of Set B "))
        for i in range(self.sizeA):
            c=int(input("enter the elements of set A "))
            self.a.append(c)
        for i in range(self.sizeB):
            d = int(input("enter the elements of set B "))
            self.b.append(d)
        print(self.a)
        print(self.b)


    def uni(self):
        uni=[]
        for element in self.a:
            if element not in uni:
                uni.append(element)
        for element in self.b:
            if element not in uni:
                uni.append(element)

        display(uni)


    def ins(self):
        ins=[]
        for element in self.a:
            for e in self.b:
                if element==e:
                    ins.append(e)
                    pass

        display(ins)


    def diff(self):
        while True:
            print("\nDifference \n"
                  "1.A-B\n"
                  "2.B-A\n"
                  "3.Previous Menu")
            ch=int(input("enter your choice "))
            if ch==1:
                diff=[]
                for element in self.a:
                    if element not in self.b:
                        diff.append(element)
                display(diff)

            elif ch==2:
                diff=[]
                for element in self.b:
                    if element not in self.a:
                        diff.append(element)
                display(diff)

            elif ch==3:
                break
            else:
                print("Choose valid choice ")


    def subset(self):
        sub=[]
        for element in self.a:
            if element  in self.b:
                sub.append(element)
        display(sub)



s1=SET()
while True:
    print("\nMenu\n"
          "1.Union\n"
          "2.Intersection\n"
          "3.Difference\n"
          "4.Subset\n"
          "5.Exit")
    ch=int(input("Enter your choice "))
    if ch==1:
        s1.uni()
    elif ch==2:
        s1.ins()
    elif ch==3:
        s1.diff()
    elif ch==4:
        s1.subset()
    elif ch==5:
        break
    else:
        print("Enter valid choice ")
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main> python -u
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main\pr1_sets.py"
enter the size of Set A 3
enter the size of Set B 2
enter the elements of set A 1
enter the elements of set A 2
enter the elements of set A 3
enter the elements of set B 4
enter the elements of set B 2
[1, 2, 3]
[4, 2]

Menu
1.Union
2.Intersection
3.Difference
4.Subset
5.Exit
Enter your choice 1
{1, 2, 3, 4}

Menu
```

```
1.Union
2.Intersection
3.Difference
4.Subset
5.Exit
Enter your choice 2
{2}

Menu
1.Union
2.Intersection
3.Difference
4.Subset
5.Exit
Enter your choice 3

Difference
1.A-B
2.B-A
3.Previous Menu
enter your choice 1
{1, 3}

Difference
1.A-B
2.B-A
3.Previous Menu
enter your choice 2
{4}

Difference
1.A-B
2.B-A
3.Previous Menu
enter your choice 3

Menu
1.Union
2.Intersection
3.Difference
4.Subset
5.Exit
Enter your choice 4
{2}

Menu
1.Union
2.Intersection
```

```
3.Difference
4.Subset
5.Exit
Enter your choice 5
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main>
```

10. Shabbir Ezzy

Title: Consider telephone book database of N clients. Make use of a hash table implementation to quickly look up client's telephone number. Make use of two collision handling techniques and compare them using number of comparisons required to find a set of telephone numbers.

```
class Hashing:
    def __init__(self):
        self.size=int(input("Enter no of phonebook users: "))
        self.table=list(None for i in range(self.size))
        self.counter=0
        self.comparison=0


    def isfull(self):
        if self.counter==self.size:
            return True
        return False


    def insert(self):
        n=0

        element=int(input("Enter element: "))
        position=element%self.size
        if self.isfull():
            print("Table is full ")

        else:
            if self.table[position] is None:
                    self.table[position]=element
                    self.counter+=1
                    self.comparison+=1

            else:
                    print("collision occured, finding new position")
                    while self.table[position] is not None:
                        position+=1
                        if position>=self.size:
                            position=0

                    self.table[position]=element
                    self.counter+=1
                    self.comparison+=1

        print(self.table[position],"appended")
        n+=1
    def display(self):
        print("Hash Table")
        for i in range(self.size):
            print(f"{i} {self.table[i]}")

    def search(self):
            element=int(input("Enter element to search: "))
            position=element%self.size

            if self.table[position]==element:
                print(f"Found at position {position}")

            else:
                while self.table[position]!=element:
                        position+=1
                        if position>=self.size:
                            position=0

                if self.table[position]==element:
                    print(f"Found at position {position}")


h1=Hashing()
choice=-1
while(choice!=0):
    print("Enter 1 to insert")
    print("Enter 2 to display hash table")
    print("Enter 3 to search an element")
    print("0:  Exit")
    choice=int(input("Enter your choice: "))

    if choice==1:
        h1.insert()

    elif choice==2:
        h1.display()

    elif choice==3:
        h1.search()

    elif choice==0:
        break
    else:
        print("Enter correct choice")
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main> python -u
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main\pr2_hash.py"
Enter no of phonebook users: 10
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 1
Enter element: 22
22 appended
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 1
Enter element: 32
collision occured, finding new position
32 appended
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 1
Enter element: 56
56 appended
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 1
Enter element: 76
collision occured, finding new position
76 appended
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 1
Enter element: 55
55 appended
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
```

```
Enter your choice: 2
Hash Table
0 None
1 None
2 22
3 32
4 None
5 55
6 56
7 76
8 None
9 None
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 3
Enter element to search: 55
Found at position 5
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 3
Enter element to search: 76
Found at position 7
Enter 1 to insert
Enter 2 to display hash table
Enter 3 to search an element
0:  Exit
Enter your choice: 0
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main>
```

10. Shabbir Ezzy

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

struct node // Node Declaration
{
    string label;
    int ch_count;
    struct node *child[10];
} * root;

class GT // Class Declaration
{
public:
    void create_tree();
    void display(node *r1);

    GT()
    {
        root = NULL;
    }
};

void GT::create_tree()
{
    int tchapters, i, j;
    root = new node;
    cout << "Enter name of book : "<<endl;
    cin.ignore();
    getline(cin, root->label);


    cout << "Enter number of chapters in book : " << endl;
    cin >> tchapters;
   // cin.ignore();
    root->ch_count = tchapters;
    cin.ignore(); // Ignore newline character after reading integer
    for (i = 0; i < tchapters; i++)
    {
        root->child[i] = new node;
        cout << "Enter the name of Chapter " << i + 1 << " : ";
        getline(cin, root->child[i]->label);
        cout << "Enter number of sections in Chapter " << root->child[i]->label << " : ";
        cin >> root->child[i]->ch_count;
        cin.ignore(); // Ignore newline character after reading integer
        for (j = 0; j < root->child[i]->ch_count; j++)
        {
            root->child[i]->child[j] = new node;
            cout << "Enter Name of Section " << j + 1 << " : ";
            getline(cin, root->child[i]->child[j]->label);
        }
    }
}

void GT::display(node *r1)
{
    int i, j;
    if (r1 != NULL)
    {
        cout << "\n-----Book Hierarchy---";
        cout << "\n Book title : " << r1->label;
        for (i = 0; i < r1->ch_count; i++)
        {
            cout << "\nChapter " << i + 1 << " : " << r1->child[i]->label;
            cout << "\nSections : ";
            for (j = 0; j < r1->child[i]->ch_count; j++)
            {
                cout << "\n" << r1->child[i]->child[j]->label;
            }
        }
    }
    cout << endl;
}

int main()
{
    int choice;
    GT gt;
    while (1)
    {
        cout << "-----------------" << endl;
        cout << "Book Tree Creation" << endl;
        cout << "-----------------" << endl;
        cout << "1.Create" << endl;
        cout << "2.Display" << endl;
        cout << "3.Quit" << endl;
        cout << "Enter your choice : ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            gt.create_tree();
            break;
        case 2:
            gt.display(root);
            break;
        case 3:
            cout << "Thanks for using this program!!!";
            exit(0);
        default:
            cout << "Wrong choice!!!" << endl;
        }
    }
    return 0;
}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if ($?) { g++
pr3_Implementation_of_General_Tree.cpp -o pr3_Implementation_of_General_Tree } ; if
($?) { .\pr3_Implementation_of_General_Tree }
-----------------
Book Tree Creation
-----------------
1.Create
2.Display
3.Quit
Enter your choice : 1
Enter name of book :
Book of Life
Enter number of chapters in book :
2
Enter the name of Chapter 1 : Prologue
Enter number of sections in Chapter Prologue : 5
Enter Name of Section 1 : 1
Enter Name of Section 2 : 2
Enter Name of Section 3 : 3
Enter Name of Section 4 : 4
Enter Name of Section 5 : 5
Enter the name of Chapter 2 : Ending
Enter number of sections in Chapter Ending : 2
Enter Name of Section 1 : Fight
Enter Name of Section 2 : End
-----------------
Book Tree Creation
-----------------
1.Create
2.Display
3.Quit
```

```
Enter your choice : 2

-----Book Hierarchy---
 Book title : Book of Life
Chapter 1 : Prologue
Sections :
1
2
3
4
5
Chapter 2 : Ending
Sections :
Fight
End
-----------------
Book Tree Creation
-----------------
1.Create
2.Display
3.Quit
Enter your choice : 3
Thanks for using this program!!!
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Title: Beginning with an empty binary search tree, Construct binary search tree by inserting the values in the order given. After constructing a binary tree:

1. Insert new node,
2. Find number of nodes in longest path from root,
3. Minimum data value found in the tree,
4. Change a tree so that the roles of the left and right pointers are swapped at every node,
5. Search a value

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
struct Tree
{
   int data;
   Tree *left;
   Tree *right;
};
class bstree
{
   public:
   Tree*create(int data)
   {
      Tree*tempTree=new Tree;
      tempTree->left=nullptr;
      tempTree->right=nullptr;
      tempTree->data=data;
      return tempTree;
   }
   void setLeft(Tree*aTree,int data)
   {
      aTree->left=create(data);
   }
   void setRight(Tree*aTree,int data)
   {
      aTree->right=create(data);
   }
   void insert(Tree*aTree,int data)
   {
      while(aTree!=NULL)
      {
         if(data<=aTree->data)
         {
            if(aTree->left!=nullptr)
            {
               aTree=aTree->left;
            }
            else
```

```
                {
                    setRight(aTree,data);
                    break;
                }
            }
            else
            {
                if(aTree->right!=nullptr)
                {
                    aTree=aTree->right;
                }
                else
                {
                    setRight(aTree,data);
                    break;
                }

            }
        }
    }

void inordertTraverse(Tree *aTree)
{
    if(aTree->left!=nullptr)
    inordertTraverse(aTree->left);
    cout<<"\n data :"<<aTree->data;
    if(aTree->right!=nullptr)
    inordertTraverse(aTree->right);


}
 int height(Tree *aTree)
    {
        int hl,hr;
        if(aTree == nullptr)
        {
            return 0;
        }
        else if(aTree->left==nullptr&& aTree->right==nullptr)
        {
            return 0;
        }

        hr=height(aTree->right);
        hl=height(aTree->left);
        if(hr>hl)
        {
            return(1+hr);
        }
        else{
            return(1+hl);
        }
    }

    void swap(Tree *aTree)
    {
        Tree *temp;
        temp=aTree;
        if(aTree !=nullptr)
        {
            swap(aTree->left);
            swap(aTree->right);
            temp=aTree->left;
            aTree->left=aTree->right;
            aTree->right=temp;
        }
    }

    int minValue(Tree *aTree)
    {
        if (aTree->left == NULL)
            return aTree->data;
        return minValue(aTree->left);
    }

    bool search(Tree *aTree,int value)
    {
        if (aTree == NULL)
        {
            return false;
        }

        while (aTree != NULL)
        {
            if (value == aTree->data) {
                return true;
            }
            else if (value < aTree->data) {
                aTree = aTree->left;
            }
            else {
                aTree = aTree->right;
            }
        }

        return false;
    }

};
int main()
{
    bstree bs;
    int ch,value;
    Tree *myTree ;
    while(ch!=8)
    {
        cout<<"\n 1. Create";
        cout<<"\n 2.insert";
        cout<<"\n 3.display";
        cout<<"\n 4.find no of nodes in largest path";
        cout<<"\n 5.find minimum value of tree";
        cout<<"\n 6.swap";
        cout<<"\n 7. Search";
        cout<<"\n 8. Exit";
        cout<<"\n Enter your choice";
        cin>>ch;
        switch (ch)
        {
            case 1:
                    cout<<"enter root";
                    cin>>value;
                    myTree = bs.create(value);
                    break;
            case 2:
                    cout<<"enter value";
                    cin>>value;
                    bs.insert(myTree,value);
                    break;

            case 3:
                    bs.inordertTraverse(myTree);
                    break;

            case 4:
                    cout<<"\n No of nodes in longest path :"<<(1+bs.height(myTree));
                    break;

            case 5:
                    cout<<"\n Minimum Data Value found in the tree is :"<<bs.minValue(myTree);
                    break;

            case 6: bs.swap(myTree);
                    cout<<"\n Tree after swaping:";
                    bs.inordertTraverse(myTree);
```

```
          break;

      case 7:cout<<"\n Enter the value to search";
          cin>> value;

          if (bs.search(myTree,value)) {
            cout << "Found "<< value<<" in tree" << endl;
          }
          else {
            cout << "Could not find "<<value <<" in tree" << endl;
          }
          break;

      default:
          cout<<"\n Enter valid input";
        break;
    }
  }


  return 0 ;
}
```

Output:

```
 PS C:\Users\Shabbir> cd "c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if
($?) { g++ pr4.cpp -o pr4 } ; if ($?) { .\pr4 }
 1. Create
 2.insert
 3.display
 4.find no of nodes in largest path
 5.find minimum value of tree
 6.swap
 7. Search
 8. Exit
 Enter your choice1
enter root34

 1. Create
 2.insert
 3.display
 4.find no of nodes in largest path
 5.find minimum value of tree
 6.swap
 7. Search
 8. Exit
 Enter your choice2
enter value23
```

```
 1. Create
 2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice2
enter value67
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice2
enter value55
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice3
data :34
data :23
data :67
data :55
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice4
No of nodes in longest path :4
1. Create
2.insert
3.display
4.find no of nodes in largest path
```

```
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice5
Minimum Data Value found in the tree is :34
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice6
Tree after swaping:
data :55
data :67
data :23
data :34
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice7
Enter the value to search23
Found 23 in tree
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
7. Search
8. Exit
Enter your choice7
Enter the value to search54
Could not find 54 in tree
1. Create
2.insert
3.display
4.find no of nodes in largest path
5.find minimum value of tree
6.swap
```

```
7. Search
8. Exit
Enter your choice8
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Title: Construct an expression tree from given prefix expression eg. +--a*bc/def and traverse it using post order traversal (non recursive) and then delete the entire tree.

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

struct node
{
    char data;
    node *left;
    node *right;
};


class tree
{
    char prefix[20];

public:
    node *top;
    void expression(char[]);
    void display(node *);
    void non_rec_postorder(node *);
    void del(node *);
};


class stack1
{
    node *data[30];
    int top;

public:
    stack1()
    {
        top = -1;
    }
    int empty()
    {
        if (top == -1)
            return 1;
        return 0;
    }
    void push(node *p)
    {
        data[++top] = p;
    }
    node *pop()
    {
        return (data[top--]);
    }
};


void tree::expression(char prefix[])
{
    char c;
    stack1 s;
    node *t1, *t2;
    int len, i;
    len = strlen(prefix);
    for (i = len - 1; i >= 0; i--)
    {
        top = new node;
        top->left = NULL;
        top->right = NULL;
        if (isalpha(prefix[i]))
        {
            top->data = prefix[i];
            s.push(top);
        }
        else if (prefix[i] == '+' || prefix[i] == '*' || prefix[i] == '-' || prefix[i] == '/')
        {
            t2 = s.pop();
            t1 = s.pop();
            top->data = prefix[i];
            top->left = t2;
            top->right = t1;
            s.push(top);
        }
    }
    top = s.pop();
}


void tree::display(node *root)
{
    if (root != NULL)
    {
        cout << root->data;
        display(root->left);
        display(root->right);
    }
```

```
}


void tree::non_rec_postorder(node *top)
{
    stack1 s1, s2;
    node *T = top;
    cout << "\n";
    s1.push(T);
    while (!s1.empty())
    {
        T = s1.pop();
        s2.push(T);
        if (T->left != NULL)
            s1.push(T->left);
        if (T->right != NULL)
            s1.push(T->right);
    }
    while (!s2.empty())
    {
        top = s2.pop();
        cout << top->data;
    }
}


void tree::del(node *node)
{
    if (node == NULL)
        return;
    del(node->left);
    del(node->right);
    cout <<endl<<"Deleting node : " << node->data<<endl;
    free(node);
}


int main()
{
    char expr[20];
    tree t;

    cout <<"Enter prefix Expression : ";
    cin >> expr;
    cout << expr;
    t.expression(expr);
   t.non_rec_postorder(t.top);
    t.del(t.top);
```

```
}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main\" ; if ($?) { g++
pr5_expression_tree_from_given_prefix_expression.cpp -o
pr5_expression_tree_from_given_prefix_expression } ; if ($?) {
.\pr5_expression_tree_from_given_prefix_expression }
Enter prefix Expression : +--a*bc/def
+--a*bc/def
abc*-de/-f+
Deleting node : a

Deleting node : b

Deleting node : c

Deleting node : *

Deleting node : -

Deleting node : d

Deleting node : e

Deleting node : /

Deleting node : -

Deleting node : f

Deleting node : +
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main>
```

10. Shabbir Ezzy

Title: Represent a given graph using adjacency matrix/list to perform DFS and using adjacency list to perform BFS. Use the map of the area around the college as the graph. Identify the prominent land marks as nodes and perform DFS and BFS on that.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
int cost[10][10], i, j, k, n, u,v;
int stk[10], top, visit1[10], visited1[10];
int main()
{
   int m;
   cout << "Enter number of vertices : ";
   cin >> n;
   cout << "Enter number of edges : ";
   cin >> m;

   cout << "\nEDGES :\n";
   for (k = 1; k <= m; k++)
   {
      cout<<"Enter U and V:";
      cin >> i >> j;
      cost[i][j] = 1;
      cost[j][i] = 1;
   }

   //display function
   cout << "The adjacency matrix of the graph is : " << endl;
   for (i = 0; i <n; i++)
   {
      for (j = 0; j <n; j++)
      {
         cout << " " << cost[i][j];
      }
      cout << endl;
   }


   cout <<endl<<"Enter initial vertex : ";
   cin >> v;
   cout << "The DFS of the Graph is\n";
   cout << v;
   visited1[v] = 1;
   k = 1;
   while (k < n)
   {
        for (j = n; j >= 1; j--)
            if (cost[v][j] != 0 && visited1[j] != 1 && visit1[j] != 1)
            {
                visit1[j] = 1;
                stk[top] = j;
                top++;
            }
        v = stk[--top];
        cout << " " << v ;
        k++;
        visit1[v] = 0;
        visited1[v] = 1;
    }
    return 0;
}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main\" ; if ($?) { g++
pr6_adjency_matrix_DFS.cpp -o pr6_adjency_matrix_DFS } ; if ($?) {
.\pr6_adjency_matrix_DFS }
Enter number of vertices : 5
Enter number of edges : 6

EDGES :
Enter U and V:1
2
Enter U and V:2
5
Enter U and V:5
4
Enter U and V:4
3
Enter U and V:3
1
Enter U and V:1
4
The adjacency matrix of the graph is :
 0 0 0 0 0
 0 0 1 1 1
 0 1 0 0 0
 0 1 0 0 1
 0 1 0 1 0
```

```
Enter initial vertex : 1
The DFS of the Graph is
1 2 5 3 4
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main>
```

10. Shabbir Ezzy

Title: You have a business with several offices; you want to lease phone lines to connect them with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve ther problem by suggesting appropriate data structures.

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


  int main()
{
        int n, i, j, k, row, col, mincost=0, min;
        char op;
        cout<<"Enter no. of vertices: ";
        cin>>n;
        int cost[n][n];
        int visit[n];
        for(i=0; i<n; i++)
                visit[i] = 0;
        for(i=0; i<n; i++)
                for(int j=0; j<n; j++)
                        cost[i][j] = -1;

        for(i=0; i<n; i++)
        {
                for(j=i+1; j<n; j++)
                {
                        cout<<"Do you want an edge between "<<i<<" and "<<j<<": ";
                        //use 'i' & 'j' if your vertices start from 0
                        cin>>op;
                        if(op=='y' || op=='Y')
                        {
                                cout<<"Enter weight: ";
                                cin>>cost[i][j];
                                cost[j][i] = cost[i][j];
                        }
                }
        }
        visit[0] = 1;
        for(k=0; k<n-1; k++)
        {
                min = 999;
                for(i=0; i<n; i++)
                {
                        for(j=0; j<n; j++)
                        {
```

```
                if(visit[i] == 1 && visit[j] == 0)
                {
                    if(cost[i][j] != -1 && min>cost[i][j])
                    {
                        min = cost[i][j];
                        row = i;
                        col = j;
                    }
                }
            }
        }
        mincost += min;
        visit[col] = 1;
        cost[row][col] = cost[col][row] = -1;
        cout<<row<<"->"<<col<<endl;
        //use 'row' & 'col' if your vertices start from 0
    }
    cout<<"\nMin. Cost: "<<mincost;
    return 0;
}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main\" ; if ($?) { g++ pr7.cpp -o pr7 } ; if
($?) { .\pr7 }
Enter no. of vertices: 5
Do you want an edge between 0 and 1: y
Enter weight: 2
Do you want an edge between 0 and 2: y
Enter weight: 7
Do you want an edge between 0 and 3: n
Do you want an edge between 0 and 4: y
Enter weight: 5
Do you want an edge between 1 and 2: y
Enter weight: 6
Do you want an edge between 1 and 3: n
Do you want an edge between 1 and 4: n
Do you want an edge between 2 and 3: y
Enter weight: 8
Do you want an edge between 2 and 4: y
Enter weight: 4
Do you want an edge between 3 and 4: y
Enter weight: 2
0->1
0->4
```

```
4->3
4->2

Min. Cost: 13
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA
Practicals\AIDS-DSA-SEM4-main\AIDS-DSA-SEM4-main>
```

10. Shabbir Ezzy

Title: Given sequence k = k1 <k2 < ... <kn of n sorted keys, with a search probability pi for each key ki . Build the Binary search tree that has the least search cost given the access probability for each key?

```
#include<iostream>
using namespace std;
#define SIZE 10
class OBST
{
int p[SIZE]; // Probabilities with which we search for an element
int q[SIZE];//Probabilities that an element is not found
int a[SIZE];//Elements from which OBST is to be built
int w[SIZE][SIZE];//Weight ‘w[i][j]’ of a tree having root
//’r[i][j]’
int c[SIZE][SIZE];//Cost ‘c[i][j] of a tree having root ‘r[i][j]
int r[SIZE][SIZE];//represents root
int n; // number of nodes
public:
/* This function accepts the input data */
void get_data()
{
int i;
cout<<"\n Optimal Binary Search Tree \n";
cout<<"\n Enter the number of nodes";
cin>>n;
cout<<"\n Enter the data as… \n";
for(i=1;i<=n;i++)
{
cout<<"\n a["<<i<<"]";
cin>>a[i];
}
for(i=1;i<=n;i++)
{
cout<<"\n p["<<i<<"]";
cin>>p[i];
}
for(i=0;i<=n;i++)
{
cout<<"\n q["<<i<<"]";
cin>>q[i];
}
}
/* This function returns a value in the range ‘r[i][j-1]’ to ‘r[i+1][j]’so
that the cost ‘c[i][k-1]+c[k][j]’is minimum */
int Min_Value(int i,int j)
{
```

```
int m,k;
int minimum=32000;
for(m=r[i][j-1];m<=r[i+1][j];m++)
{
if((c[i][m-1]+c[m][j])<minimum)
{
minimum=c[i][m-1]+c[m][j];
k=m;
}
}
return k;
}
/* This function builds the table from all the given probabilities It
basically computes C,r,W values */
void build_OBST()
{
int i,j,k,l,m;
for(i=0;i<n;i++)
{
//initialize
w[i][i]=q[i];
r[i][i]=c[i][i]=0;
//Optimal trees with one node
w[i][i+1]=q[i]+q[i+1]+p[i+1];
r[i][i+1]=i+1;
c[i][i+1]=q[i]+q[i+1]+p[i+1];
}
w[n][n]=q[n];
r[n][n]=c[n][n]=0;
//Find optimal trees with ‘m’ nodes
for(m=2;m<=n;m++)
{
for(i=0;i<=n-m;i++)
{
j=i+m;
w[i][j]=w[i][j-1]+p[j]+q[j];
k=Min_Value(i,j);
c[i][j]=w[i][j]+c[i][k-1]+c[k][j];
r[i][j]=k;
}
}
}
/* This function builds the tree from the tables made by the OBST function */
void build_tree()
{
int i,j,k;
int queue[20],front=-1,rear=-1;
cout<<"The Optimal Binary Search Tree For the Given Node Is…\n";
```

```
cout<<"\n The Root of this OBST is ::"<<r[0][n];
cout<<"\nThe Cost of this OBST is::"<<c[0][n];
cout<<"\n\n\t NODE \t LEFT CHILD \t RIGHT CHILD ";
cout<<"\n";
queue[++rear]=0;
queue[++rear]=n;
while(front!=rear)
{
i=queue[++front];
j=queue[++front];
k=r[i][j];
cout<<"\n\t"<<k;
if(r[i][k-1]!=0)
{
cout<<"\t\t"<<r[i][k-1];
queue[++rear]=i;
queue[++rear]=k-1;
}
else
cout<<"\t\t";
if(r[k][j]!=0)
{
cout<<"\t"<<r[k][j];
queue[++rear]=k;
queue[++rear]=j;
}
else
cout<<"\t";
}//end of while
cout<<"\n";
}
};//end of the class
/*This is the main function */
int main()
{
OBST obj;
obj.get_data();
obj.build_OBST();
obj.build_tree();
return 0;
}
```

Output:

```
PS C:\Users\Shabbir> cd "c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if
($?) { g++ pr8.cpp -o pr8 } ; if ($?) { .\pr8 }

 Optimal Binary Search Tree

 Enter the number of nodes4

 Enter the data asΓÇª

 a[1]34

 a[2]12

 a[3]20

 a[4]56

 p[1]2

 p[2]4

 p[3]1

 p[4]3

 q[0]2

 q[1]4

 q[2]1

 q[3]3

 q[4]5
The Optimal Binary Search Tree For the Given Node IsΓÇª

 The Root of this OBST is ::2
The Cost of this OBST is::51

        NODE    LEFT CHILD      RIGHT CHILD

       2            1      4
       1
       4            3
       3
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Title: A Dictionary stores keywords and its meanings. Provide facility for adding new keywords, deleting keywords, updating values of any entry. Provide a facility to display whole data sorted in ascending/ Descending order. Also find how many maximum comparisons may require for finding any keyword. Use Height balance tree and find the complexity for finding a keyword.

```
#include<iostream>
#include<cstring>
#include<cstdlib>
#define MAX 50
#define SIZE 20
using namespace std;

struct AVLnode
{
   public:
   char cWord[SIZE],cMeaning[MAX];
   AVLnode *left,*right;
   int iB_fac,iHt;
};

class AVLtree
{

   public:
      AVLnode *root;
      AVLtree()
      {
         root=NULL;
      }
      int height(AVLnode*);
      int bf(AVLnode*);
      AVLnode* insert(AVLnode*,char[SIZE],char[MAX]);
      AVLnode* rotate_left(AVLnode*);
      AVLnode* rotate_right(AVLnode*);
      AVLnode* LL(AVLnode*);
      AVLnode* RR(AVLnode*);
      AVLnode* LR(AVLnode*);
      AVLnode* RL(AVLnode*);
      AVLnode* delet(AVLnode*,char x[SIZE]);
      void inorder(AVLnode*);
};

AVLnode *AVLtree::delet(AVLnode *curr,char x[SIZE])
{
   AVLnode *temp;
```

```
    if(curr==NULL)
        return(0);
    else
        if(strcmp(x,curr->cWord)>0)
        {
            curr->right=delet(curr->right,x);
            if(bf(curr)==2)
            if(bf(curr->left)>=0)
                curr=LL(curr);
            else
                curr=LR(curr);
        }
        else
        if(strcmp(x,curr->cWord)<0)
        {
            curr->left=delet(curr->left,x);
            if(bf(curr)==-2)
            if(bf(curr->right)<=0)
                curr=RR(curr);
            else
                curr=RL(curr);
        }
    else
    {
        if(curr->right!=NULL)
        {
            temp=curr->right;
            while(temp->left!=NULL)
            temp=temp->left;
            strcpy(curr->cWord,temp->cWord);
            curr->right=delet(curr->right,temp->cWord);
            if(bf(curr)==2)
            if(bf(curr->left)>=0)
                curr=LL(curr);
            else
                curr=LR(curr);
        }
        else
        return(curr->left);
    }
    curr->iHt=height(curr);
    return(curr);
}


AVLnode* AVLtree :: insert(AVLnode*root,char newword[SIZE],char newmeaning[MAX])
{
    if(root==NULL)
    {
        root=new AVLnode;
        root->left=root->right=NULL;
        strcpy(root->cWord,newword);
        strcpy(root->cMeaning,newmeaning);
    }

    else if(strcmp(root->cWord,newword)!=0)
    {
        if(strcmp(root->cWord,newword)>0)
        {
            root->left=insert(root->left,newword,newmeaning);
            if(bf(root)==2)
            {
                if (strcmp(root->left->cWord,newword)>0)
                    root=LL(root);
                else
                    root=LR(root);
            }
        }

        else if(strcmp(root->cWord,newword)<0)
        {
            root->right=insert(root->right,newword,newmeaning);
            if(bf(root)==-2)
            {
                if(strcmp(root->right->cWord,newword)>0)
                    root=RR(root);
                else
                    root=RL(root);
            }
        }
    }
    else
        cout<<"\nRedundant AVLnode";
    root->iHt=height(root);
    return root;
}

int AVLtree :: height(AVLnode* curr)
{
    int lh,rh;
    if(curr==NULL)
        return 0;
    if(curr->right==NULL && curr->left==NULL)
        return 0;
    else
    {
        lh=lh+height(curr->left);
        rh=rh+height(curr->right);
        if(lh>rh)
            return lh+1;
        return rh+1;
    }
}

int AVLtree :: bf(AVLnode* curr)
{
    int lh,rh;
    if(curr==NULL)
        return 0;
    else
    {
        if(curr->left==NULL)
            lh=0;
        else
            lh=1+curr->left->iHt;
        if(curr->right==NULL)
            rh=0;
        else
            rh=1+curr->right->iHt;
        return(lh-rh);
    }
}

AVLnode* AVLtree :: rotate_right(AVLnode* curr)
{
    AVLnode* temp;
    temp=curr->left;
    curr->left=temp->right;
    temp->left=curr;
    curr->iHt=height(curr);
    temp->iHt=height(temp);
    return temp;
}

AVLnode* AVLtree :: rotate_left(AVLnode* curr)
{
    AVLnode* temp;
    temp=curr->right;
    curr->right=temp->left;
    temp->left=curr;
    curr->iHt=height(curr);
    temp->iHt=height(temp);
    return temp;
}
```

```
AVLnode* AVLtree :: RR(AVLnode* curr)
{
    curr=rotate_left(curr);
    return curr;
}

AVLnode* AVLtree :: LL(AVLnode* curr)
{
    curr=rotate_right(curr);
    return curr;
}

AVLnode* AVLtree :: RL(AVLnode* curr)
{
    curr->right=rotate_right(curr->right);
    curr=rotate_left(curr);
    return curr;
}

AVLnode* AVLtree::LR(AVLnode* curr)
{
    curr->left=rotate_left(curr->left);
    curr=rotate_right(curr);
    return curr;
}

void AVLtree :: inorder(AVLnode* curr)
{
    if(curr!=NULL)
    {
        inorder(curr->left);
        cout<<"\n\t"<<curr->cWord<<"\t"<<curr->cMeaning;
        inorder(curr->right);
    }
}

int main()
{
    int iCh;
    AVLtree a;
    AVLnode *curr=NULL;
    char cWd[SIZE],cMean[MAX];
    cout<<"\n-------------------------------------";
    cout<<"\n\tAVL TREE IMPLEMENTATION";
    cout<<"\n-------------------------------------";
    do
    {   cout<<"\n--------------------------------";
```

```
        cout<<"\n\t\tMENU";
        cout<<"\n--------------------------------";
        cout<<"\n1.Insert\n2.Inorder\n3.Delete\n4.Exit";
        cout<<"\n--------------------------------";
        cout<<"\nEnter your choice :";
        cin>>iCh;

        switch(iCh)
        {
          case 1: cout<<"\nEnter Word : ";
            cin>>cWd;
            cout<<"\nEnter Meaning : ";
            cin.ignore();
            cin.getline(cMean,MAX);

            a.root=a.insert(a.root,cWd,cMean);
            break;

          case 2: cout<<"\n\tWORD\tMEANING";
            a.inorder(a.root);
            break;

          case 3: cout<<"\nEnter the word to be deleted : ";
              cin>>cWd;
              curr=a.delet(a.root,cWd);
              if(curr==NULL)
                cout<<"\nWord not present!";
              else
                cout<<"\nWord deleted Successfully!";
              curr=NULL;
              break;

          case 4: exit(0);
        }
    }while(iCh!=4);

    return 0;
}
```

Output:

```
PS C:\Users\Shabbir> cd "c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if
($?) { g++ pr9.cpp -o pr9 } ; if ($?) { .\pr9 }

-------------------------------------
      AVL TREE IMPLEMENTATION
-------------------------------------
--------------------------------
            MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :1

Enter Word : a

Enter Meaning : apple

--------------------------------
            MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :1

Enter Word : b

Enter Meaning : banana

--------------------------------
            MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :2

      WORD    MEANING
      a       apple
      b       banana
```

```
--------------------------------
             MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :1

Enter Word : a

Enter Meaning : avenger

Redundant AVLnode
--------------------------------
             MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :2

       WORD    MEANING
       a       apple
       b       banana
--------------------------------
             MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :3

Enter the word to be deleted : a

Word deleted Successfully!
--------------------------------
             MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
```

```
--------------------------------
Enter your choice :2

        WORD    MEANING
        b       apple
--------------------------------
                MENU
--------------------------------
1.Insert
2.Inorder
3.Delete
4.Exit
--------------------------------
Enter your choice :4
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Consider a scenario for Hospital to cater services to different kinds of patients as

a) Serious (top priority),
b) non-serious (medium priority),
c) General Check-up (Least priority).
Implement the priority queue to cater services to the patients.

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

string Q[10];
int pr[10];
int r=-1,f=-1,n;

void enqueue(string data, int p){
  int i;
  if((f==0)&&(r==n-1))
  cout<<"Queue is full";
  else{
    if(f==-1){
      f=r=0;
      Q[r]=data;
      pr[r]=p;
      }
      else {
        for(i=r;i>=f;i--){
          if(p>pr[i]){
            Q[i+1]=Q[i];
            pr[i+1]=pr[i];
          }
          else break;
        }

      Q[i+1]=data;
      pr[i+1]=p;
      r++;
  }
}
}

void dequeue(){
  if(f==-1){
    cout<<"Queue is Empty";
  }
  else{
        cout<<"Element deleted = "<<Q[f]<<endl;
        cout<<"Element Priority = "<<pr[f]<<endl;
        if(f==r) f=r=-1;
        else f++;
    }
}

void print(){
    int i;
    for(i=f;i<=r;i++){
        cout<<"\nPatient Name: "<<Q[i];
        switch(pr[i]){
            case 1: cout<<" Priority: ketchup";
            break;
            case 2: cout<<" Priority: non-serious";
            break;
            case 3: cout<<" Priority: highly-serious";
            break;
            default:
            cout<<" Priority Not Found";
        }
    }

}

int main(){
    string data;
    int opt,i,p;


    do{
        cout<<"\n\n1. insert data in queue\n2. show data of the queue\n3. delete data from the
queue\n4. Exit\n";
        cout<<"Enter your choice : ";
        cin>>opt;
        switch (opt)
        {
        case 1:
            cout<<"enter no of patients"<<endl;
            cin>>n;
            for(i=0;i<n;i++){
                cout<<"Enter Patient Name: ";
                cin>>data;
                cout<<"Enter Priority: ";
                cin>>p;
                enqueue(data,p);
            }
            break;
    case 2:
      print();
      break;

    case 3:
      dequeue();
      break;
    }
  }while (opt!=4);
  return 0;
  }
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if ($?) { g++ pr10.cpp -o pr10 }
; if ($?) { .\pr10 }


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 1
enter no of patients
4
Enter Patient Name: Shabbir
Enter Priority: 2
Enter Patient Name: aditya
Enter Priority: 3
Enter Patient Name: tongale
Enter Priority: 1
Enter Patient Name: zoman
Enter Priority: 2


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 2

Patient Name: aditya Priority: highly-serious
Patient Name: Shabbir Priority: non-serious
Patient Name: zoman Priority: non-serious
Patient Name: tongale Priority: ketchup
```

```
1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 3
Element deleted = aditya
Element Priority = 3


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 2

Patient Name: Shabbir Priority: non-serious
Patient Name: zoman Priority: non-serious
Patient Name: tongale Priority: ketchup

1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 3
Element deleted = Shabbir
Element Priority = 2


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 3
Element deleted = zoman
Element Priority = 2


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 2

Patient Name: tongale Priority: ketchup

1. insert data in queue
2. show data of the queue
```

```
3. delete data from the queue
4. Exit
Enter your choice : 3
Element deleted = tongale
Element Priority = 1


1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 2

Patient Name: ` Priority Not Found

1. insert data in queue
2. show data of the queue
3. delete data from the queue
4. Exit
Enter your choice : 4
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Title: Department maintains student information. The file contains roll number, name, division and address. Allow users to add, delete information about students. Display information of a particular employee. If the record of the student does not exist an appropriate message is displayed. If it is, then the system displays the student details. Use a sequential file to maintain the data.

```
#include<iostream>
#include<fstream>
#include<cstring>
#include<stdlib.h>

using namespace std;



class Student
{
  typedef struct studentinfo
   {
      char name[50];
      int rollno;
      char division[5];
      char address[100];
   }rec ;
   rec records;

   public:

   void create();
   void display();
   void search();
   void Delete(int a);

};

void Student:: create()
{
   char ch ='y';
   fstream seq;
   seq.open("StudentRecord.txt",ios::out);
   do{
      cout << "Enter name : ";
      cin >> records.name;

      cout << "Enter roll number : ";
      cin >> records.rollno;

      cout << "Enter division : ";
      cin >> records.division;

      cout << "Enter address : ";
      cin >> records.address;

      seq.write((char*)&records,sizeof(records));

       cout <<"\n Do you want to add more records :  ";
       cin >> ch;
   }while (ch=='y');


   seq.close();

}

void Student::display()
{
   fstream seq;
   int n;
   seq.open("StudentRecord.txt",ios::in);
   seq.seekg(0,ios::beg);
   cout << "\n Content of file are ... "<< endl;
    while (seq.read((char*)&records, sizeof(records)))
    {
      if(records.rollno!=-1)
      {
          cout << "\nName: " << records.name;
          cout << "\nRoll No: " << records.rollno;
          cout << "\nDivision: " << records.division;
          cout << "\nAddress: " << records.address << endl;
      }
   }
   int lastrecord = seq.tellg();
   n = lastrecord/(sizeof(rec));
}
void Student::search()

{
   fstream seq;
   int id,pos;
   cout << "\n Enter the roll number to search";
   cin >> id;
  seq.open("StudentRecord.txt",ios::in|ios::binary);
  seq.seekg(0,ios::beg);
  bool found = false;
   while (seq.read((char*)&records, sizeof(records)))

   {
      if(records.rollno==id)
      {
          found= true;
          cout<<"Student record found";
          cout << "\nRoll Number: " << records.rollno << endl;
          cout << "\nName: " << records.name << endl;
          cout << "\nDivision: " << records.division << endl;
          cout << "\nAddress: " << records.address << endl;
          break;
      }


   }
   seq.close();
   if(!found)
   {
      cout << "Roll No :"<< id << " is not found!" << endl;
   }

}

void Student::Delete(int id)
{

   ifstream infile;
   ofstream outfile;
   infile.open("StudentRecord.txt",ios::in);//open file for read purpose
   outfile.open("temp.txt",ios::app);//create file for write purpose if no matching record found
   infile.seekg(0,ios::beg);
   bool flag =false;
   while(infile.read((char *)&records,sizeof(records)))
   {
      if(records.rollno==id)
      {
          flag =true;
          continue;
      }
      outfile.write((char *)&records,sizeof(records));

   }
    infile.close();
    outfile.close();

    if(flag==false)
    {   remove("temp.txt");
        cout<<"\nRoll no :"<< id <<" is not present in record.";

    }
    else
    {
        remove("StudentRecord.txt");
        rename("temp.txt","StudentRecord.txt");
        cout << "Record deleted successfully." ;

    }


}


int main()
{
    Student s;
    char ans ='y';
    int ch,id;
    bool key;
    do
    {
        cout << "\n 1. Create";
        cout << "\n 2. Display";
        cout << "\n 3. search";
        cout << "\n 4. Delete";
        cout << "\n 5.exit";

        cout << "\n Enter your choice ";
        cin >> ch;

        switch(ch)
        {
             case 1: s.create();
                    break;

            case 2: s.display();
                    break;

            case 3: s.search();
                    break;
```

```
           case 4: cout << "enter the roll no to delete";
                   cin>> id;
                   s.Delete(id);
                   break;


           default:
           cout << "\n Enter valid choice";
           break;

       }
        cout << "\n Do you want to go back to main menu";
        cin >> ans;

    }while(ans == 'y');
    return 0;
}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if ($?) { g++ pr11.cpp -o pr11 }
; if ($?) { .\pr11 }

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 1
Enter name : Shabbir
Enter roll number : 39
Enter division : d
Enter address : nashik

 Do you want to add more records :  y
Enter name : aditya
Enter roll number : 66
Enter division : e
Enter address : nashik

 Do you want to add more records :  y
Enter name : tongale
Enter roll number : 61
Enter division : e
Enter address : vapi
```

```
 Do you want to add more records :  y
Enter name : zoman
Enter roll number : 68
Enter division : a
Enter address : dindori

 Do you want to add more records :  n

 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 2

 Content of file are ...

Name: Shabbir
Roll No: 39
Division: d
Address: nashik

Name: aditya
Roll No: 66
Division: e
Address: nashik

Name: tongale
Roll No: 61
Division: e
Address: vapi

Name: zoman
Roll No: 68
Division: a
Address: dindori

 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 3
 Enter the roll number to search39
Student record found
Roll Number: 39

Name: Shabbir

Division: d

Address: nashik

 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 3

 Enter the roll number to search66
Student record found
Roll Number: 66

Name: aditya

Division: e

Address: nashik

 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 4
enter the roll no to delete39
Record deleted successfully.
 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 2
 Content of file are ...

Name: aditya
Roll No: 66
Division: e
Address: nashik

Name: tongale
Roll No: 61
Division: e
Address: vapi

Name: zoman
Roll No: 68
Division: a
Address: dindori

 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 4
enter the roll no to delete68
Record deleted successfully.
 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 2

 Content of file are ...

Name: aditya
Roll No: 66
Division: e
Address: nashik

Name: tongale
Roll No: 61
Division: e
Address: vapi
```

```
 Do you want to go back to main menuy

 1. Create
 2. Display
 3. search
 4. Delete
 5.exit
 Enter your choice 5

 Enter valid choice
 Do you want to go back to main menun
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```

10. Shabbir Ezzy

Title: Company maintains employee information as employee ID,
name, designation and salary. Allow users to add, delete
information of employees. Display information of a
particular employee. If an employee does not exist an
appropriate message is displayed. If it is, then the
system displays the employee details. Use index sequential
file to maintain the data.

```
#include<iostream>
#include<fstream>
#include <sstream>
#include <string>

using namespace std;

class employee
{
   typedef struct empinfo
   {
      int empid;
      char empName[50];
      char empDesignation[50];
      float empSalary;

   } rec;
   rec records;

public :
   void create();
   void Delete(int id);
   void display();
   void print();

};
void employee::print()
{
   cout << "------------Details of Employee---------" << endl;
   cout << "EMployee ID :" << records.empid << std::endl;
   cout << "EMployee Name :" << records.empName<< std::endl;
   cout << "EMployee Designation :" << records.empDesignation<< std::endl;
   cout << "EMployee Salary :" << records.empSalary<< std::endl;
}


static int findEmployeePosition(int employeeID)
{
    ifstream indexFile("index.txt");
    if (!indexFile)
    {
        cout << "Error opening index file." << endl;
        return -1;
    }

    string line;
    while (getline(indexFile, line))
     {
        istringstream iss(line);
        int id, position;
        if (iss >> id >> position)
        {
            if (id == employeeID)
            {
                indexFile.close();
                return position;
            }
        }
    }

    indexFile.close();
    return -1;
}
void employee::create()
{
    fstream file;
    fstream indexfile;
    char ch = 'y';

    file.open("employee.txt", ios::app);
    indexfile.open("index.txt", ios::app);

    do
    {
        cout << "Enter employee Id : ";
        cin >> records.empid;

        cout << "Enter employee Name : ";
        cin >> records.empName;

        cout << "Enter employee Designation : ";
        cin >> records.empDesignation;

        cout << "Enter employee salary : ";
        cin >> records.empSalary;
        // Get the current position (offset) in the data file
        int position = file.tellp();

        file.write((char*)& records, sizeof(records));


        // Write employee ID and file offset to the index file
        indexfile << records.empid << " " << position << endl;

        cout << "Employee added successfully." << endl;

        cout << "\n Do you want to add more records :  ";
        cin >> ch;

    } while (ch=='y');
    file.close();
    indexfile.close();

}
void employee::display()
{
    int empId = -1;
    cout << "Enter employee Id : ";
    cin >> empId;
    if (empId>0)
    {
        int pos = findEmployeePosition(empId);
        if (pos < 0)
        {
            cout << "No matching Employee Record available " << endl;

            return;
        }
        else
        {
            fstream file;
            file.open("employee.txt", ios::in);
            file.seekg(pos);
            file.read((char*)& records, sizeof(records));

            print();
            file.close();
        }
    }


}
void employee::Delete(int employeeId)
{
```

```
// Open the index file
fstream indexFile("index.txt",ios::in);
if (!indexFile) {
    cout << "Error opening index file." << endl;
    return;
}

// Open the data file
fstream file("employee.txt", ios::in);
if (!file) {
    cout << "Error opening data file." << endl;
    indexFile.close();
    return;
}

// Create a temporary file to store updated index entries
ofstream tempIndexFile("tempIndex.txt",ios::out);
if (!tempIndexFile) {
    cout << "Error creating temporary index file." << endl;
    indexFile.close();
    file.close();
    return;
}

string line;
int id;
int position;
bool found = false;
// Read each line from the index file
while (getline(indexFile, line)) {
    istringstream iss(line);
    if (iss >> id >> position) {
        if (id == employeeId) {
            found = true;
            // Skip the record by not writing it to the temporary index file
            continue;
        }
    }

    // Write the index entry to the temporary index file
    tempIndexFile << line << endl;
}

// Close the files
indexFile.close();
file.close();
tempIndexFile.close();
if (!found)
```

```
    {
        remove("tempIndexFile.txt");
        cout << "Employee record not found" << endl;
    }
    else
    {
        remove("index.txt");
        rename("tempIndex.txt", "index.txt");

        cout << "Employee deleted successfully." << endl;

    }
    // Remove the original index file and rename the temporary index file

}
int main()
{
    employee emp;
    int employeeId;
    int choice;
    char ans = 'y';
    do
    {
        cout << "1. Add Employee" << endl;
        cout << "2. Delete Employee" << endl;
        cout << "3. Display Employee" << endl;
        cout << "5. Exit" << endl;

        cout << " \n Enter your choice" << endl;
        cin >> choice;

        switch (choice)
        {
        case 1: emp.create();
            break;

        case 2:cout << "Enter employee Id to delete : " << endl;
            cin >> employeeId;
            emp.Delete(employeeId);
            break;

        case 3:
          emp.display();
            break;
        case 4:
            return 0;
        default:
            cout << "Enter  valid choice" << endl;
            break;
        }
    } while (true);

    return 0;

}
```

Output:

```
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals> cd
"c:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals\" ; if ($?) { g++ pr12.cpp -o pr12 }
; if ($?) { .\pr12 }
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
1
Enter employee Id : 1
Enter employee Name : Shabbir
Enter employee Designation : engineer
Enter employee salary : 35000
Employee added successfully.

 Do you want to add more records :  y
Enter employee Id : 2
Enter employee Name : aditya
Enter employee Designation : trainer
Enter employee salary : 26000
Employee added successfully.

 Do you want to add more records :  n
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
3
Enter employee Id : 1
------------Details of Employee---------
EMployee ID :1
EMployee Name :Shabbir
EMployee Designation :engineer
EMployee Salary :35000
```

```
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
3
Enter employee Id : 2
------------Details of Employee---------
EMployee ID :2
EMployee Name :aditya
EMployee Designation :trainer
EMployee Salary :26000
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
2
Enter employee Id to delete :
1
Employee deleted successfully.
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
3
Enter employee Id : 1
No matching Employee Record available
1. Add Employee
2. Delete Employee
3. Display Employee
5. Exit

 Enter your choice
3
Enter employee Id : 2
------------Details of Employee---------
EMployee ID :2
EMployee Name :aditya
EMployee Designation :trainer
EMployee Salary :26000
1. Add Employee
2. Delete Employee
3. Display Employee
```

```
5. Exit

 Enter your choice
5
PS C:\Users\Shabbir\Desktop\Shabbir\aids SE\DSA Practicals>
```