## Practical No.1

**Title:** Consider telephone book database of N clients. Make use of a hash table implementation to quickly look up a 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:** SE COMPUTER ENGINEERING

**Roll No.:** 45

### Date of Performance : 09/03/2023 Date of Submission : 10/03/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

class HashingDemo:

def \_\_init\_\_(self):

self.size=int(input("enter the size of the hash tabe : ")) self.HashTable=list(0 for i in range(self.size)) self.num\_of\_elements = 0 self.comparisons=0

def isTableFull(self):

if self.num\_of\_elements==self.size:

return True

else:

return False

def HashFun(self,element):

return element % self.size

def insertElement\_Linear(self,element):

if self.isTableFull(): print("Hash Table is Full") return False OccupiedStatus=False position=self.HashFun(element) if self.HashTable[position]==0:

self.HashTable[position]=element print("Telephone Number " + str(element) + " at position

" +str(position))

OccupiedStatus= True self.num\_of\_elements+=1 else:

print("Collision has occured for Telephone number " +

str(element) + " at index " + str(position)) position= self.LinearProbing(element, position) self.HashTable[position]=element OccupiedStatus=True self.num\_of\_elements +=1

return OccupiedStatus

def LinearProbing(self,element,position):

while self.HashTable[position] !=0: position +=1 if position >=self.size:

position=0

return position

def InsertElement\_Quadratic(self, element):

if self.isTableFull(): print("Hash Table Full ") return False OccupiedStatus= False position= self.HashFun(element) if self.HashTable[position]==0:

self.HashTable[position]=element print("Telephone Number " + str(element) + " at position

" +str(position))

OccupiedStatus= True self.num\_of\_elements+=1

else:

print("Collision has occured for Telephone number " +

str(element) + " at index " + str(position))

OccupiedStatus, position= self.quadraticProbing(element,

position) if OccupiedStatus:

self.HashTable[position]=element self.num\_of\_elements +=1

self.HashTable[position]=element OccupiedStatus = True self.num\_of\_elements+= 1

return OccupiedStatus

def quadraticProbing(self,element,position): Found=False limit=50 i=1 while i<=limit:

newPosition = position + (i\*\*2) newPosition = newPosition % self.size if self.HashTable[newPosition]==0: Found=True break else:

i+=1

return Found,newPosition

def search(self,element):

found=False position=self.HashFun(element) self.comparisons += 1 if(self.HashTable[position]==element):

return position isFound=True else:

temp=position-1 while position<self.size:

if self.HashTable[position]!=element:

position +=1 self.comparisons+=1

else: return position

position=temp while position >=0:

if self.HashTable[position]!=element:

position -=1 self.comparisons +=1

else: return position if not found: print("Element not found") return False

def display(self):

print("\n") print("-------------------------------------------") print("\n Position \t Telephone Number\n") print("-------------------------------------------") for i in range(self.size):

print("\t"+ str(i) + "====>" + str(self.HashTable[i]))

Hash\_object1=HashingDemo() print("\nInserting the telephone numbers in the hash table...\n") print("\nCollision Resolution using Linear Probing\n")

Hash\_object1.insertElement\_Linear(1111111112)

Hash\_object1.insertElement\_Linear(3333333331)

Hash\_object1.insertElement\_Linear(4444444417)

Hash\_object1.insertElement\_Linear(5555555590)

Hash\_object1.insertElement\_Linear(6666666621)

Hash\_object1.insertElement\_Linear(7777777788) Hash\_object1.insertElement\_Linear(8888888840)

Hash\_object1.insertElement\_Linear(9999999977) Hash\_object1.display() print() print("The position of number 3333333331 is : " + str(Hash\_object1.search(3333333331))) print("The position of number 6666666621 is : " + str(Hash\_object1.search(6666666621))) print("The position of number 9999999977 is : " + str(Hash\_object1.search(9999999977))) print("-------------------------------------------") print("\nTotal Number of comparisons done for searching : "+ str(Hash\_object1.comparisons)) print("-------------------------------------------") print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\ n")

hash\_object2=HashingDemo() print("\nInserting the telephone numbers in the hash table...\n") print("\nCollision Resolution using quadratic Probing\n") hash\_object2.InsertElement\_Quadratic(1111111112) hash\_object2.InsertElement\_Quadratic(3333333331) hash\_object2.InsertElement\_Quadratic(4444444417) hash\_object2.InsertElement\_Quadratic(5555555590) hash\_object2.InsertElement\_Quadratic(6666666621) hash\_object2.InsertElement\_Quadratic(7777777788) hash\_object2.InsertElement\_Quadratic(8888888840) hash\_object2.InsertElement\_Quadratic(9999999977) hash\_object2.display() print() print("The position of number 4444444417 is : " + str(hash\_object2.search(4444444417))) print("The position of number 5555555590 is : " + str(hash\_object2.search(5555555590))) print("The position of number 7777777788 is : " + str(hash\_object2.search(7777777788))) print("-------------------------------------------") print("\nTotal Number of comparisons done for searching : "+ str(hash\_object2.comparisons)) print("-------------------------------------------") print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\ n")

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ python3 DSA1.py enter the size of the hash tabe : 10

Inserting the telephone numbers in the hash table...

Collision Resolution using Linear Probing

Telephone Number 1111111112 at position 2

Telephone Number 3333333331 at position 1

Telephone Number 4444444417 at position 7

Telephone Number 5555555590 at position 0

Collision has occured for Telephone number 6666666621 at index 1 Telephone Number 7777777788 at position 8

Collision has occured for Telephone number 8888888840 at index 0

Collision has occured for Telephone number 9999999977 at index 7

-------------------------------------------

Position Telephone Number

-------------------------------------------

0====>5555555590

1====>3333333331

2====>1111111112

3====>6666666621

4====>8888888840

5====>0

6====>0

7====>4444444417

8====>7777777788

9====>9999999977

The position of number 3333333331 is : 1

The position of number 6666666621 is : 3

The position of number 9999999977 is : 9

-------------------------------------------

Total Number of comparisons done for searching : 7

-------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* enter the size of the hash tabe : 5

Inserting the telephone numbers in the hash table...

Collision Resolution using quadratic Probing

Telephone Number 1111111112 at position 2

Telephone Number 3333333331 at position 1

Collision has occured for Telephone number 4444444417 at index 2

Telephone Number 5555555590 at position 0

Hash Table Full

Hash Table Full

Hash Table Full

Hash Table Full

-------------------------------------------

Position Telephone Number

-------------------------------------------

0====>5555555590

1====>3333333331

2====>1111111112

3====>4444444417

4====>0

The position of number 4444444417 is : 3

The position of number 5555555590 is : 0

Element not found

The position of number 7777777788 is : False

-------------------------------------------

Total Number of comparisons done for searching : 9 -------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.2

**Title:**For given set of elements create skip list. Find the element in the set that is closest to some given value. (note: Decide the level of element in the list

Randomly with some upper limit)

**Class:** SE COMPUTER ENGINEERING

**Roll No.:** 45

### Date of Performance : 16/03/2023 Date of Submission : 21/03/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

import random class Node(object):

def \_\_init\_\_(self, key,level): self.key=key self.next=[None]\*(level+1)

class SkipList(object):

def \_\_init\_\_(self,MaxLevel,Fraction):

self.Upper\_Limit=MaxLevel self.Fraction=Fraction self.header=self.CreateNode(self.Upper\_Limit,-1) self.level=0

def CreateNode(self,LevelNo,key):

n=Node(key,LevelNo) return n

def randomLevel(self): LevelNo=0

while random.random()<self.Fraction and LevelNo<self.Upper\_Limit:

LevelNo+=1 return LevelNo

def InsertElement(self,key):

Data=[None]\*(self.Upper\_Limit+1) CurrentNode=self.header for i in range(self.level,-1,-1):

while CurrentNode.next[i] and CurrentNode.next[i].key <

key:

CurrentNode=CurrentNode.next[i]

Data[i]=CurrentNode

CurrentNode=CurrentNode.next[0]

if CurrentNode==None or CurrentNode.key!=key:

rlevel=self.randomLevel() if rlevel>self.level:

for i in range(self.level+1,rlevel+1):

Data[i]=self.header self.level=rlevel

temp=self.CreateNode(rlevel,key) for i in range (rlevel+1): temp.next[i]=Data[i].next[i] Data[i].next[i]=temp print("Inserting Key {} ",format(key)) def searchElement(self,key): CurrentNode=self.header for i in range(self.level,-1,-1):

while CurrentNode.next[i] and CurrentNode.next[i].key<key:

CurrentNode=CurrentNode.next[i] CurrentNode=CurrentNode.next[0] if CurrentNode and CurrentNode.key==key:

print("The key ",key," has nearest node

",CurrentNode.next[i].key) def display(self): print("\nThe Skip List is as follows...") head=self.header for LevelNo in range(self.level+1): print("Level{",format(LevelNo),"} : ",end="") node=head.next[LevelNo] while(node !=None):

print(node.key,end=" ") node=node.next[LevelNo]

print("")

def main():

print("\n Enter Upper Limit For Level ") upperLimit=int(input()) lst=SkipList(upperLimit,0.5) lst.InsertElement(5) lst.InsertElement(8) lst.InsertElement(10) lst.InsertElement(12) lst.InsertElement(15) lst.InsertElement(18) lst.InsertElement(24) lst.InsertElement(27) lst.InsertElement(28) lst.InsertElement(30) lst.display() lst.searchElement(15) lst.searchElement(27)

main()

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ python3 DSA2.py

Enter Upper Limit For Level

5

Inserting Key {} 5

Inserting Key {} 8

Inserting Key {} 10

Inserting Key {} 12

Inserting Key {} 15

Inserting Key {} 18

Inserting Key {} 24

Inserting Key {} 27

Inserting Key {} 28

Inserting Key {} 30

The Skip List is as follows...

Level{ 0 } : 5 8 10 12 15 18 24 27 28 30

Level{ 1 } : 10 24 28 30

Level{ 2 } : 24 28 30

Level{ 3 } : 24

The key 15 has nearest node 18 The key 27 has nearest node 28 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.3

**Title**:A book consists of chapters, chapters consist of sections and sections consist of subsections. Construct a tree and print the nodes. Find the time and space requirements of your method.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 23/03/2023 **Date of Submission :** 28/03/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<iostream> using namespace std; class TREE\_CLASS

{ private: typedef struct bin

{ char data[50]; struct bin \*left; struct bin \*right;

}node;

public:

node \*New,\*root; TREE\_CLASS(); void create(); void insert(node \*,node \*); void rec\_inorder(node \*); void printLevelOrder(node\* root); void printLevel(node\* root,int level); int height(node\* node);

};

TREE\_CLASS::TREE\_CLASS() { root = NULL;

} void TREE\_CLASS::create() { char ans='y'; do

{

New=new node; cout<<"\nEnter the elemnets="; cin>>New->data; New->left=NULL;

New->right=NULL; if(root==NULL) root=New; else insert(root,New); cout<<"\nDo you want to insert more elements?(y/n)="; cin>>ans;

}while(ans=='y'||ans=='y');

}

void TREE\_CLASS::insert(node \*root,node \*New)

{ char ch; cout<<"\nWhere to insert left/right of(l/r)->"<<root->data<<":"; cin>>ch;

if((ch=='r')||(ch=='R'))

{ if(root->right==NULL)

{ root->right=New;

} else

insert(root->right,New);

} else

{ if(root->left==NULL)

{ root->left=New;

} else insert(root->left,New);

} }

void TREE\_CLASS::rec\_inorder(node \*root)

{ if(root!=NULL)

{

rec\_inorder(root->left); cout<<""<<root->data; rec\_inorder(root->right);

}

} void TREE\_CLASS::printLevelOrder(node\* root)

{ int h=height(root); int i; for(i=1;i<=h;i++)

{ cout<<"\n"; printLevel(root,i);

}

}

void TREE\_CLASS::printLevel(node\* root,int level)

{ if(root==NULL) return; if(level==1)

cout<<""<<root->data;

else if(level>1)

{ printLevel(root->left,level-1); printLevel(root->right,level-1);

}

}

int TREE\_CLASS::height(node\* node)

{ if(node==NULL) return 0; else

{ int lheight=height(node->left); int rheight=height(node->right);

if(lheight>rheight) return(lheight+1); else return(rheight+1); }

}

int main()

{ int choice; TREE\_CLASS obj;

do

{ cout<<"\nMAIN MENU"; cout<<"\n1.create"; cout<<"\n2.display"; cout<<"\n3.Exit"; cout<<"\nEnter your choice:"; cin>>choice; switch(choice)

{ case 1: obj.create(); break;

case 2: if(obj.root==NULL) cout<<"TREE IS NOT CREATED"; else obj.printLevelOrder(obj.root);

break; }

}while(choice<=2); return 0;

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA3.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

MAIN MENU

1.create

2.display

3.Exit

Enter your choice:1

Enter the elemnets=5

Do you want to insert more elements?(y/n)=y

Enter the elemnets=4

Where to insert left/right of(l/r)->5:l

Do you want to insert more elements?(y/n)=y

Enter the elemnets=6

Where to insert left/right of(l/r)->5:r

Do you want to insert more elements?(y/n)=y

Enter the elemnets=3

Where to insert left/right of(l/r)->5:l

Where to insert left/right of(l/r)->4:l

Do you want to insert more elements?(y/n)=y

Enter the elemnets=7

Where to insert left/right of(l/r)->5:r

Where to insert left/right of(l/r)->6:r

Do you want to insert more elements?(y/n)=n

MAIN MENU

1.create

2.display

3.Exit

Enter your choice:2

5

46

37

MAIN MENU

1.create

2.display

3.Exit

Enter your choice:3 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.4

**Title:**Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

**Class:** SE COMPUTER ENGINEERING

**Roll No.:** 45

### Date of Performance : 11/04/2023 Date of Submission : 13/04/2023

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<iostream> using namespace std;

struct TTREE

{ struct TTREE \*lc; char lt; char data; char rt; struct TTREE \*rc;

}; typedef struct TTREE ttree;

class myttree

{ private : ttree \*head;

public :

myttree(); void create\_btree(); void insert\_btree(ttree \*);

void thread\_btree(); void display\_ttree(); void display\_btree();

};

myttree :: myttree()

{

head = NULL;

}

void myttree :: insert\_btree(ttree \*node)

{ if(head == NULL)

head = node;

else {

int flag = 0; char ans; ttree \*par; par = head; while(flag == 0)

{ cout<<"\nWhere to add (l/r) of "<<par->data<< " : "; cin>>ans; if(ans == 'l')

{ if(par->lc == NULL)

{ par->lc = node; flag = 1;

} else

par = par->lc;

} else { if(par->rc == NULL)

{ par->rc = node; flag = 1;

} else

par = par->rc;

}

}

} }

void myttree :: create\_btree()

{ int i,n; ttree \*node; cout<<"\nEnter the total no. of nodes in btree : "; cin>>n;

for(i = 1; i<=n; i++)

{ node = new ttree; node->lc = NULL; node->lt = 'f'; node->rc = NULL; node->rt = 'f'; cout<<"\nEnter the data field of node "<<i<<" : "; cin>>node->data; insert\_btree(node);

} cout<<"\nBinary tree created successfully";

}

void preorder(ttree \*node, ttree \*A[],int & n)

{ if(node != NULL) {

A[n++] = node; preorder(node->lc,A,n); preorder(node->rc,A,n);

} }

void myttree :: thread\_btree()

{ ttree \*A[20]; ttree \*node; int n = 0,i; if(head == NULL)

{ cout<<"\nBinary tree is empty";

} else { preorder(head,A,n); cout<<"\nPreorder traversal is : "; for(i = 0 ; i < n ;i++)

{ cout<<A[i]->data<<" ";

} node = new ttree; node->lt = 'f'; node->lc = head; node->data = '\0'; node->rt = 't'; node->rc = node; head = node; for(i = 0 ; i < n ;i++)

{ if(A[i]->lc == NULL)

{ if(i ==0)

A[i]->lc = head; else

A[i]->lc = A[i-1];

A[i]->lt = 't';

} if(A[i]->rc == NULL)

{

if(i == n - 1)

A[i]->rc = head; else

A[i]->rc = A[i+1];

A[i]->rt = 't';

} } cout<<"\nBinary Tree Threaded successfully\n";

} }

void myttree :: display\_ttree()

{ if(head == NULL)

cout<<"\nBinary tree is empty";

else { cout<<"Preorder Traversal of the tree : "; ttree \*temp = head->lc; while( temp != head)

{ while(temp->lt == 'f')

{ cout<<temp->data<<" "; temp = temp->lc;

} cout<<temp->data<<" "; while(temp->rt == 't' && temp->rc != head)

{ temp = temp->rc; cout<<temp->data<<" ";

} if(temp->lt == 'f') temp = temp->lc; else

temp = temp->rc;

}

} }

void inorder(ttree \*node)

{ if(node != NULL)

{ inorder(node->lc); cout<<node->data<<" "; inorder(node->rc);

} }

void myttree :: display\_btree()

{ if(head == NULL)

cout<<"\nBinary tree is empty";

else { cout<<"Inorder Traversal of the tree : "; inorder(head);

} } int main()

{ int ch; myttree t1;

do { cout<<"\n\t1: Create Binary Tree"; cout<<"\n\t2: Display Binary Tree"; cout<<"\n\t3: Thread the Tree "; cout<<"\n\t4: Display the threaded tree"; cout<<"\n\t5: Exit"; cout<<"\n\nEnter ur choice : "; cin>>ch; switch(ch) { case 1 : t1.create\_btree(); break; case 2 : t1.display\_btree();

break;

case 3 : t1.thread\_btree(); break;

case 4 : t1.display\_ttree(); break;

case 5 : cout<<"\nend\n"; break;

default: cout<<"\nTry again\n";

} }while(ch != 5); return 0;

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA4.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

1: Create Binary Tree

2: Display Binary Tree

3: Thread the Tree

4: Display the threaded tree

5: Exit

Enter ur choice : 1

Enter the total no. of nodes in btree : 4

Enter the data field of node 1 : 5

Enter the data field of node 2 : 4

Where to add (l/r) of 5 : l

Enter the data field of node 3 : 6

Where to add (l/r) of 5 : r

Enter the data field of node 4 : 6

Where to add (l/r) of 5 : r

Where to add (l/r) of 6 : l

Binary tree created successfully

1: Create Binary Tree

2: Display Binary Tree

3: Thread the Tree

4: Display the threaded tree

5: Exit

Enter ur choice : 2

Inorder Traversal of the tree : 4 5 6 6

1: Create Binary Tree

2: Display Binary Tree

3: Thread the Tree

4: Display the threaded tree

5: Exit

Enter ur choice : 3

Preorder traversal is : 5 4 6 6

Binary Tree Threaded successfully

1: Create Binary Tree

2: Display Binary Tree

3: Thread the Tree

4: Display the threaded tree

5: Exit

Enter ur choice : 4

Preorder Traversal of the tree : 5 4 6 6

1: Create Binary Tree

2: Display Binary Tree

3: Thread the Tree

4: Display the threaded tree

5: Exit Enter ur choice : 5

end prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.5

**Title:**A Dictionary stores keywords & its meanings. Provide facility for adding new keywords, deleting keywords, & updating values of any entry. Also provide facility to display whole data sorted in ascending/ Descending order, also find how many maximum comparisons may require for finding any keyword. Use Binary Search Tree for implementation.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 18/04/2023 **Date of Submission :** 20/04/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

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

class node

{

public: char key[20];

char meaning[20]; node \*left,\*right;

node() { left=right=NULL;

} };

class dictionary

{ public:

node \*root; dictionary() { root=NULL;

} void addkey(); void insert(node\*); void display\_asc(); void inorder(node\*); void display\_desc(); void rtorder(node\*); void findkey(); void updatekey(); void deletekey(); node\* bstsearch(node \*,char[]); void bstdelete(node\*,node\*); node\* parent(node\*,node\*); node \*findlargest(node\*);

};

void dictionary::addkey()

{ node \*temp; int n,i;

cout<<"\nEnter no of keys to be inserted: "; cin>>n;

for(i=0;i<n;i++) { temp=new node(); cout<<"\nEnter Keyword : "; cin>>temp->key; cout<<"\nEnter Meaning : "; cin>>temp->meaning; insert(temp);

} }

void dictionary::display\_asc()

{ if(root==NULL) cout<<"\nDictionary is Empty"; else inorder(root);

} void dictionary::display\_desc()

{ if(root==NULL) cout<<"\nDictionary is Empty"; else rtorder(root);

} void dictionary::inorder(node \*r)

{ if(r!=NULL)

|  |  |
| --- | --- |
| { inorder(r->left); |  |
| cout<<"\n"<<r->key<<" : = inorder(r->right);  } } void dictionary::rtorder(node \*r)  { if(r!=NULL) { rtorder(r->right); | "<<r->meaning; |
| cout<<"\n"<<r->key<<" : = rtorder(r->left);  }  } | "<<r->meaning; |

void dictionary::insert(node \*temp)

{ node \*temp1,\*temp2;

if(root==NULL) { root=temp;

} else { temp1=root; while(temp1!=NULL) { temp2=temp1; if(strcmp(temp->key,temp1->key)<0) temp1=temp1->left; else temp1=temp1->right;

} if(strcmp(temp->key,temp2->key)<0) temp2->left=temp; else temp2->right=temp;

} }

void dictionary::findkey()

{ char key[20]; node \*temp; cout<<"\nEnter key to be searched: "; cin>>key; temp=bstsearch(root,key); if(temp==NULL) cout<<"\nKey is not present in the Dictionary"; else

{ cout<<"\nKey Found..!!!";

cout<<"\n"<<temp->key<<" : "<<temp->meaning; } } void dictionary::updatekey()

{ char key[20]; node \*temp;

cout<<"\nEnter key to be Updated: "; cin>>key; temp=bstsearch(root,key); if(temp==NULL) cout<<"\nKey is not present in the Dictionary"; else

{ cout<<"\nEnter New meaning for "<<temp->key; cin>>temp->meaning; cout<<"\n"<<temp->key<<" : "<<temp->meaning; } }

void dictionary::deletekey()

{ char key[20]; node \*temp1,\*temp2,\*largest; cout<<"\nEnter key to be deleted: "; cin>>key; temp1=bstsearch(root,key); if(temp1==NULL) cout<<"\nKey is not Present in dictionary"; else { cout<<"\nKey deleted Successfully..!!"; if(temp1->left==NULL&&temp1==root) { root=temp1->right; delete temp1;

} else if(temp1==root&&temp1->left!=NULL)

{ largest=findlargest(temp1->left); strcpy(temp1->key,largest->key); strcpy(temp1->meaning,largest->meaning); if(temp1->left==largest) { temp1->left=NULL;

} else { temp2=parent(temp1->left,largest); bstdelete(largest,temp2);

} } else { temp2=parent(root,temp1); bstdelete(temp1,temp2);

}

} } void dictionary::bstdelete(node \*temp1,node \*temp2)

{ node \*largest;

if(strcmp(temp1->key,temp2->key)<0) { if(temp1->left==NULL) { temp2->left=temp1->right; delete temp1;

} else if(temp1->right==NULL)

{ temp2->left=temp1->right; delete temp1;

} else { largest=findlargest(temp1->left); strcpy(temp1->key,largest->key); strcpy(temp1->meaning,largest->meaning); temp2=parent(temp1->left,largest); bstdelete(largest,temp2);

}

} else { if(temp1->left==NULL) { temp2->right=temp1->right; delete temp1;

} else if(temp1->right==NULL)

{ temp2->right=temp1->right; delete temp1;

} else { largest=findlargest(temp1->left); strcpy(temp1->key,largest->key); strcpy(temp1->meaning,largest->meaning); temp2=parent(temp1->left,largest); bstdelete(largest,temp2);

}

} } node \*dictionary::parent(node \*r,node\*temp)

{ node \*temp1,\*temp2; temp1=r; while(temp1!=temp) { temp2=temp1; if(strcmp(temp->key,temp1->key)<0) temp1=temp1->left; else temp1=temp1->right;

} return temp2;

} node \*dictionary::findlargest(node \*t)

{ if(t->right==NULL) return t; else return findlargest(t->right);

}

node \*dictionary::bstsearch(node \*r,char key[20])

{ if(r==NULL) return NULL; if(strcmp(key,r->key)<0) return bstsearch(r->left,key); else if(strcmp(key,r->key)>0) return bstsearch(r->right,key); else return r;

} int main()

{ int choice; char ch; dictionary dct; do { cout<<"\*\*\*\*\*\* MENU \*\*\*\*\*"; cout<<"\n1. Add keys to Dictionary"; cout<<"\n2. Display Dictionary in Ascending order"; cout<<"\n3. Display Dictionary in Descending order"; cout<<"\n4. Find Key"; cout<<"\n5. Update Key"; cout<<"\n6. Delete Key"; cout<<"\n\nEnter your choice: "; cin>>choice; switch(choice) { case 1: dct.addkey(); break;

case 2: dct.display\_asc(); break;

case 3: dct.display\_desc(); break;

case 4: dct.findkey(); break;

case 5: dct.updatekey(); break;

case 6: dct.deletekey(); break;

default:cout<<"\n\nInvalid Choie....!!!";

} cout<<"\n\nDo you want to continue..[y/n] ? : "; cin>>ch;

}while(ch!='n'); return 0;

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA5.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 1

Enter no of keys to be inserted: 2

Enter Keyword : Hello

Enter Meaning : hi

Enter Keyword : Nope

Enter Meaning : no

Do you want to continue..[y/n] ? : y

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 2

Hello : = hi

Nope : = no

Do you want to continue..[y/n] ? : y

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 3

Nope : = no Hello : = hi

Do you want to continue..[y/n] ? : y

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 4

Enter key to be searched: Nope

Key Found..!!! Nope : no

Do you want to continue..[y/n] ? : y

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 5

Enter key to be Updated: Nope

Enter New meaning for NopeNoo

Nope : Noo

Do you want to continue..[y/n] ? : y

\*\*\*\*\*\* MENU \*\*\*\*\*

1. Add keys to Dictionary
2. Display Dictionary in Ascending order
3. Display Dictionary in Descending order
4. Find Key
5. Update Key
6. Delete Key

Enter your choice: 6 Enter key to be deleted: Hello Key deleted Successfully..!!

Do you want to continue..[y/n] ? : n prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.6

**Title:**There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight take to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representation used.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 23/04/2023 **Date of Submission :** 23/04/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include <iostream>

#include <list>

#include <algorithm> #define MAX\_NODES 50 #define inf 10000 using namespace std;

class Node { public:

int city; float dist;

Node(int C, float D)

{

city = C; dist = D;

}

~Node() {}

};

list<Node\*> adj[MAX\_NODES]; int totNodes; const int WHITE=0, BLACK=1; int colour[MAX\_NODES]; float dist1[MAX\_NODES];

int pie[MAX\_NODES];

void CreateGraph()

{ int neighbours, C; float D; Node\* nNew; list<Node\*> l1; list<Node\*>::iterator it; cout << "Enter total no. of cities in Tour : "; cin >> totNodes; for (int i=1; i<=totNodes; i++)

{

cout << "Enter no. of neighbours of City " << i << ": "; cin >> neighbours; for (int j=1; j<=neighbours; j++)

{

cout << "Neighbour #" << j << " City no: "; cin >> C; cout << "Distance: "; cin >> D; nNew = new Node(C, D); l1.push\_back(nNew);

} adj[i] = l1; l1.clear();

} } void DisplayGraph()

{ list<Node\*>::iterator it; for (int i=1; i<=totNodes; i++)

{ for (it = adj[i].begin(); it != adj[i].end(); it++) cout << " (" << i << ")----" << ((\*it)->dist) << "----(" << (\*it)->city << ")" << endl; } } void print\_path(int s, int d)

{

if (d == 0)

{ cout << "\nNo path!\n"; return;

}

if (d == s) cout << " " << s;

else

{ print\_path(s, pie[d]); cout << "---->" << d;

} } void shortestPath(int source, int destination)

{ float dij,z; int p = 0;

list<Node\*>::iterator it; for (int i1=1; i1<=totNodes; i1++)

{ dist1[i1] = inf; colour[i1] = WHITE;

} dist1[source] = 0; int i = source; // Latest vertex permanently labeled colour[source] = BLACK; pie[source] = 0; while (p != destination)

{ int m = inf; // smallest distance for (int j=1; j<=totNodes; j++)

{ if (colour[j]==BLACK) continue; else { it = adj[i].begin(); while (it != adj[i].end())

{ if ((\*it)->city == j) break; else it++;

} if (it==adj[i].end()) dij = inf; else dij = (\*it)->dist; z = dij + dist1[i]; if (z > inf) z = inf; if (z < dist1[j])

{ dist1[j] = z; pie[j] = i;

} if (dist1[j] < m)

{ m = dist1[j]; p = j;

}

} } colour[p] = BLACK; i = p;

} cout << "\nThe shortest Distance from vertex " << source << " to

vertex " << destination << " is "

<< dist1[destination] << endl; if (pie[destination]==0) cout << "\nNo Path from " << destination <<

" to " << source << ".\n"; cout << "The Path is as follows:\n\n"; print\_path(source, destination); cout << endl;

} int main()

{ list<Node\*>::iterator it; cout << "\*\*\*\*\*\*\*\*\*\*\*\*Dijkstra's Shortest Path Algorithm \*\*\*\*\*\*\*\*\*\*\*"

<< endl;

CreateGraph(); DisplayGraph(); int sNode,dNode; cout << "Enter Starting Point: "; cin >> sNode; cout << "Enter Destination (different than source): "; cin >> dNode; shortestPath(sNode, dNode); for (int i1=1; i1<=totNodes; i1++)

{ for (it = adj[i1].begin(); it != adj[i1].end(); it++) delete \*it;

}

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA6.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

\*\*\*\*\*\*\*\*\*\*\*\*Dijkstra's Shortest Path Algorithm \*\*\*\*\*\*\*\*\*\*\*

Enter total no. of cities in Tour : 3

Enter no. of neighbours of City 1: 2

Neighbour #1 City no: 1 Distance: 23

Neighbour #2 City no: 2

Distance: 33

Enter no. of neighbours of City 2: 2

Neighbour #1 City no: 3 Distance: 43

Neighbour #2 City no: 4

Distance: 34

Enter no. of neighbours of City 3: 1

Neighbour #1 City no: 5

Distance: 45

(1)----23----(1)

(1)----33----(2)

(2)----43----(3)

(2)----34----(4)

(3)----45----(5)

Enter Starting Point: 1

Enter Destination (different than source): 2

The shortest Distance from vertex 1 to vertex 2 is 33 The Path is as follows:

1---->2 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.7

**Title:**You have a business with several offices; you want to lease phone lines to connect them up 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 the problem by suggesting appropriate data structures.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 25/04/2023 **Date of Submission :** 27/04/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include <iostream> #include <stdlib.h> using namespace std;

int cost[10][10], i, j, k, n, qu[10], front, rear, v, visit[10], visited[10]; 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++)

{

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 << "Enter initial vertex : "; cin >> v; cout << "The BFS of the Graph is\n"; cout << v<<endl; visited[v] = 1; k = 1; while (k < n)

{

for (j = 1; j <= n; j++) if (cost[v][j] != 0 && visited[j] != 1 && visit[j] != 1)

{ visit[j] = 1; qu[rear++] = j;

} v = qu[front++]; cout << v << " ";

k++; visit[v] = 0; visited[v] = 1;

}

cout <<endl<<"Enter initial vertex : "; cin >> v; cout << "The DFS of the Graph is\n"; cout << v<<endl; visited[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:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA7.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

Enter number of vertices : 4

Enter number of edges : 5

EDGES :

1

2

1

3

2

3

3

4

4

1

The adjacency matrix of the graph is :

0 0 0 0

0 0 1 1

0 1 0 1

0 1 1 0

Enter initial vertex : 1

The BFS of the Graph is

1

2 3 4

Enter initial vertex : 1

The DFS of the Graph is

1

2 1 3

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.8

**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?

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 27/04/2023 **Date of Submission :** 02/05/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<iostream> #define SIZE 10 using namespace std;

class OBST

{ private:

int p[SIZE]; int q[SIZE]; int a[SIZE]; int w[SIZE][SIZE]; int c[SIZE][SIZE]; int r[SIZE][SIZE]; int n; int front,rear,queue[20];

public:

OBST(); void get\_data(); int Min\_Value(int,int); void OBST1(); void build\_tree();

};

OBST::OBST()

{ front=rear=-1;

}

void OBST::get\_data()

{

int i; cout<<"\nOptimal Binary Search Tree\n"; cout<<"\nEnter the number of nodes::"; cin>>n; cout<<"\nEnter the data as....\n"; for(i=1;i<=n;i++) { cout<<"\na["<<i<<"]:"; cin>>a[i];

} cout<<"\nEnter the Probabilities for successful searches::"; for(i=1;i<=n;i++) { cout<<"\np["<<i<<"]:"; cin>>p[i];

} cout<<"\nEnter the Probabilities for unsuccessful searches::"; for(i=0;i<=n;i++) { cout<<"\nq["<<i<<"]:"; cin>>q[i];

} }

int OBST::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;

}

void OBST::OBST1()

{ int i,j,k,m; for(i=0;i<n;i++) { w[i][i]=q[i]; r[i][i]=c[i][i]=0; 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; 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;

}

} }

void OBST::build\_tree()

{ int i,j,k;

cout<<"\nThe Optimal Binary Search Tree For The Given Nodes

Is......."; cout<<"\nThe root of OBST is:: "<<r[0][n]; cout<<"\nThe Cost of this OBST is::"<<c[0][n]; cout<<"\n\n\n\tNODE\tLEFT CHILD\tRIGHT CHILD"; cout<<"\n------------------------------------ "<<endl; 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<<" "<<r[i][k-1];

queue[++rear]=i; queue[++rear]=k-1;

} else

cout<<" -";

if(r[k][j]!=0)

{

cout<<" "<<r[k][j];

queue[++rear]=k; queue[++rear]=j;

} else

cout<<" -";

} cout<<endl;

}

int main()

{ OBST obj; obj.get\_data(); obj.OBST1(); obj.build\_tree(); return 0;

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA8.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

Optimal Binary Search Tree Enter the number of nodes::4 Enter the data as....

a[1]:1 a[2]:2 a[3]:3 a[4]:4

Enter the Probabilities for successful searches:: p[1]:1 p[2]:2

p[3]:3 p[4]:4

Enter the Probabilities for unsuccessful searches:: q[0]:1 q[1]:2 q[2]:3 q[3]:4 q[4]:5

The Optimal Binary Search Tree For The Given Nodes Is.......

The root of OBST is:: 3

The Cost of this OBST is::51

NODE LEFT CHILD RIGHT CHILD

------------------------------------

3 2 4

2 1 -

4 - -

1 - -

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.9

**Title:**A Dictionary stores keywords & its meanings. Provide facility for adding new keywords, deleting keywords, updating values of any entry. Provide 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.

**Class:** SE COMPUTER ENGINEERING

# **Roll No.:** 45

**Date of Performance :** 02/05/2023 **Date of Submission :** 09/05/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<iostream>

#include<stdlib.h> #include<string.h> using namespace std; class avlnode

{ public:

char keyword[20]; char meaning[30]; int ht; avlnode \*left; avlnode \*right;

avlnode() { left=right=NULL;

} };

class dictionary

{ public: avlnode \*root; void addkeyword(); avlnode\* place\_keyword(avlnode\*,avlnode\*); avlnode \*LL(avlnode\*); avlnode \*RR(avlnode\*); avlnode \*RL(avlnode\*); avlnode \*LR(avlnode\*);

avlnode \*rotateright(avlnode\*); avlnode \*rotateleft(avlnode\*); int balance(avlnode \*); int height(avlnode\*); void display\_asc(); void inorder(avlnode \*); void display\_dsc(); void rtorder(avlnode \*); void search\_keyword(); void update\_keyword(); avlnode \*avlsearch(avlnode\*,char[]); dictionary() { root=NULL;

} };

void dictionary::addkeyword()

{ int no,i; avlnode \*temp;

cout<<"\nEnter no of Keywords : "; cin>>no;

for(i=0;i<no;i++) { temp=new avlnode(); cout<<"\nEnter Keyword : "; cin>>temp->keyword; cout<<"\nEnter meaning : "; cin>>temp->meaning; root=place\_keyword(root,temp);

} }

avlnode\* dictionary::place\_keyword(avlnode\* r,avlnode \*temp)

{ if(r==NULL) { r=temp;

} else if(strcmp(temp->keyword,r->keyword)<0)

{ r->left=place\_keyword(r->left,temp); if(balance(r)==2)

{

if(strcmp(temp->keyword,r->left->keyword)<0) { r=LL(r);

} else { r=LR(r);

}

} } else if(strcmp(temp->keyword,r->keyword)>0)

{ r->right=place\_keyword(r->right,temp); if(balance(r)==-2) { if(strcmp(temp->keyword,r->right->keyword)>0) { r=RR(r);

} else { r=RL(r);

}

} } r->ht=height(r); return r;

}

avlnode\* dictionary::LL(avlnode \*temp)

{ temp=rotateright(temp); return temp;

}

avlnode\* dictionary::RR(avlnode \*temp)

{ temp=rotateleft(temp); return temp;

}

avlnode\* dictionary::RL(avlnode \*temp)

{ temp->right=rotateright(temp->right); temp=rotateleft(temp); return temp;

}

avlnode\* dictionary::LR(avlnode \*temp)

{ temp->left=rotateleft(temp->left); temp=rotateright(temp); return temp;

}

avlnode\* dictionary::rotateright(avlnode \*temp)

{ avlnode \*y; y=temp->left; temp->left=y->right; y->right=temp; temp->ht=height(temp); y->ht=height(y); return y; }

avlnode\* dictionary::rotateleft(avlnode \*temp)

{ avlnode \*y; y=temp->right; temp->right=y->left; y->left=temp; temp->ht=height(temp); y->ht=height(y); return y; }

int dictionary::balance(avlnode \*temp)

{ int lh,rh;

if(temp==NULL) return(0); if(temp->left==NULL) lh=0; else lh=(temp->left->ht)+1;

if(temp->right==NULL) rh=0; else rh=(temp->right->ht)+1; return(lh-rh);

}

int dictionary::height(avlnode \*temp)

{ int lh,rh; if(temp==NULL) return(0); if(temp->left==NULL) lh=0; else lh=(temp->left->ht)+1;

if(temp->right==NULL) rh=0; else rh=(temp->right->ht)+1;

if(lh>rh) return (lh); else return(rh);

}

void dictionary::display\_asc()

{

cout<<"\n\n Keyword Meaning";

cout<<"\n---------------------------------"; inorder(root);

}

void dictionary::inorder(avlnode \*r)

{ if(r!=NULL) { inorder(r->left); cout<<"\n"<<r->keyword<<" "<<r->meaning; inorder(r->right);

} }

void dictionary::display\_dsc()

{ cout<<"Keyword"<<" "<<"Meaning";

cout<<"\n---------------------------------"; rtorder(root);

}

void dictionary::rtorder(avlnode \*temp)

{ if(temp!=NULL) { rtorder(temp->right); cout<<"\n"<<temp->keyword<<" "<<temp->meaning; rtorder(temp->left);

}

}

void dictionary::search\_keyword()

{ char targetkey[20]; avlnode \*temp;

cout<<"\nEnter Keyword To Be Searched :"; cin>>targetkey; temp=avlsearch(root,targetkey);

if(temp==NULL) { cout<<"\nKeyword Is Not Present In The Dictionary..!!..";

} else { cout<<"\n"<<temp->keyword<<" "<<temp->meaning; } }

void dictionary::update\_keyword()

{ char targetkey[20]; avlnode \*temp;

cout<<"\nEnter Keyword To Be updated :"; cin>>targetkey; temp=avlsearch(root,targetkey);

if(temp==NULL) { cout<<"\nKeyword Is Not Present In The Dictionary..!!..";

} else { cout<<"\n"<<temp->keyword<<" "<<temp->meaning; cout<<"\nEnter New Meaning"; cin>>temp->meaning; cout<<"\nYour Meaning has been Updated"; } }

avlnode \* dictionary::avlsearch(avlnode \* temp,char targetkey[20])

{ if(temp==NULL) { return NULL;

} if(strcmp(targetkey,temp->keyword)<0) return avlsearch(temp->left,targetkey); else if(strcmp(targetkey,temp->keyword)>0) return avlsearch(temp->right,targetkey); else return temp;

}

int main()

{ dictionary dict; int choice;

while(1) { cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*"; cout<<"\n1. Add Keyword"; cout<<"\n2. Display Dictionary in Ascending order"; cout<<"\n3. Display in Descending Order"; cout<<"\n4. Search Keyword"; cout<<"\n5. Update Keyword In The Dictionary."; cout<<"\n6. Exit";

cout<<"\n\nEnter your choice: "; cin>>choice;

switch(choice) { case 1: dict.addkeyword(); break; case 2: dict.display\_asc(); break;

case 3: dict.display\_dsc(); break;

case 4: dict.search\_keyword(); break;

case 5: dict.update\_keyword(); break;

case 6: exit(0);

default: cout<<"\n\nInvalid Choice: ";

}

}

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA9.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 1

Enter no of Keywords : 2

Enter Keyword : Hello

Enter meaning : hi

Enter Keyword : NOpe

Enter meaning : No

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 2

Keyword Meaning

---------------------------------

Hello hi

NOpe No

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 3

Keyword Meaning

---------------------------------

NOpe No

Hello hi

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 4

Enter Keyword To Be Searched :Hello Hello hi

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 5

Enter Keyword To Be updated :heloo

Keyword Is Not Present In The Dictionary..!!..

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword
2. Display Dictionary in Ascending order
3. Display in Descending Order
4. Search Keyword
5. Update Keyword In The Dictionary.
6. Exit

Enter your choice: 6 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.10

**Title:**Read the marks obtained by second year in an online examination of a particular subject. Find out maximum and minimum marks obtained in that subject. Use heap data structure. Analyze the algorithm.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 09/05/2023 **Date of Submission :** 16/05/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<iostream> using namespace std; #define MAX 10 class Heap

{ private: int Marks[MAX]; int n;

public:

Heap(); void insert(int num); void makeheap(); void heapsort(); void display(); void display\_max\_min\_marks();

};

Heap::Heap()

{ n=0;

for(int i=0;i<MAX;i++) Marks[i]=0; }

void Heap::insert(int num)

{ if(n<MAX) { Marks[n]=num; n++;

} else cout<<"\nArray is full...";

}

void Heap::makeheap()

{ for(int i=1;i<n;i++)

{ int val=Marks[i]; int j=1; int f=(j-1)/2; while((j>0) && Marks[f]<val)

{ Marks[j]=Marks[f]; j=f; f=(j-1)/2;

}

Marks[j]=val;

} }

void Heap::heapsort()

{ for(int i=n-1;i>0;i--)

{ int temp=Marks[i]; Marks[i]=Marks[0]; int k=0; int j; if(i==1) j=-1; else j=1; if(i>2&&Marks[2]>Marks[1]) j=2;

while(j>=0&&temp<Marks[j]) { Marks[k]=Marks[j]; k=j; j=2\*k+1; if(j+1<=i-1&&Marks[j]<Marks[j+1]) j++; if(j>i-1) j=-1;

}

Marks[k]=temp;

} }

void Heap::display()

{ for(int i=0;i<n;i++) cout<<" "<<Marks[i];

cout<<"\n";

}

void Heap::display\_max\_min\_marks()

{ cout<<"\nThe Maximum marks="<<Marks[n-1]; cout<<"\nThe Minimum marks="<<Marks[0]; cout<<"\n";

}

int main()

{

Heap obj; obj.insert(55); obj.insert(48); obj.insert(89); obj.insert(91); obj.insert(75); obj.insert(63); obj.insert(45); obj.insert(78); cout<<"\nFollowing Marks are obtained by students...."<<endl; obj.display(); obj.makeheap(); cout<<"\n\nHeapified....."<<endl; obj.heapsort(); obj.display\_max\_min\_marks(); return 0;

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA10.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

Following Marks are obtained by students....

55 48 89 91 75 63 45 78

Heapified.....

The Maximum marks=91 The Minimum marks=45 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$

### Practical No.11

**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.

**Class:** SE COMPUTER ENGINEERING

### Roll No.: 45

**Date of Performance :** 16/05/2023 **Date of Submission :** 18/05/2023 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Program :**

#include<fstream>

#include<stdio.h>

#include<string.h>

#include<iostream>

#include<stdlib.h> #include<iomanip> using namespace std; class file { private:

int roll; char name[100]; char Div[20]; char addr[20];

public:

void input(); void show(); int getn()

{ return roll;

} };file fileobj;

void file::input()

{ cout<<"Enter the rollno,name,Div and address :"; cin>>roll>>name>>Div>>addr;

//gets(name);

} void file::show()

{

cout<<setiosflags(ios::right)<<setw(10) <<roll<<setiosflags(ios::right)<<setw(10)

<<name<<setiosflags(ios::left)<<setw(5)<<Div

<<setw(10)<<addr<<endl; } void Create(); void Add(); void Display(); void DisplayP(); void Modify(); void Delete();

fstream fil; int main()

{ int opt;

while(1)

{ cout<<"1.Create Data File"<<endl; cout<<"2.Add New Record in Data File"<<endl; cout<<"3.Display Record From Data File"<<endl; cout<<"4.Display Particular Record From Data File"<<endl; cout<<"5.Modify Paricular Record From Data File"<<endl; cout<<"6.Delete Particular Record From Data File"<<endl; cout<<"7.Exit From the Program"<<endl; cout<<"Enter your Option : "<<endl; cin>>opt; switch(opt) { case 1:

{ Create(); cout<<"Display Main Menu"<<endl; getchar(); break;

} case 2:

{ Add(); cout<<"Display Main Menu"<<endl; getchar(); break;

}

case 3:

{ cout<<"\n\*\*\*\*\*Student Information\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

cout<<"\t"<<"RollNo"<<"\t"<<"Name"<<"\t"<<"Div"<<"\t"<<"Address"<<endl;

Display(); cout<<"Display Main Menu"<<endl; getchar(); break;

} case 4:

{ DisplayP(); cout<<"Display Main Menu"<<endl; getchar(); break;

} case 5:

{ Modify(); cout<<"Display Main Menu"<<endl; getchar(); break;

} case 6:

{ Delete(); cout<<"Display Main Menu"<<endl;

getchar(); break;

} case 7:

{ exit(0);

} default: { cout<<"Wrong Choice....Press Key For View the Main

Menu"; getchar();//getch(); //clrscr();

}

}

}

}

void Create() //Function to Create Data File

{

char ch='y'; fil.open("binary.DAT",ios::out| ios::binary); while(ch=='y' || ch =='Y')

{ fileobj.input(); fil.write((char\*)&fileobj, sizeof(fileobj)); cout<<"Want to Continue....."; cin>>ch;

} fil.close();

}

void Add() //Function to Add New Record in Data File

{ char ch='y'; fil.open("binary.DAT",ios::app| ios::binary); while(ch=='y' || ch =='Y')

{ fileobj.input(); fil.write((char\*)&fileobj, sizeof(fileobj)); cout<<"Want to Continue....."; cin>>ch;

} fil.close();

}

void Display() //Function to Display All Record from Data File

{ fil.open("binary.DAT",ios::in| ios::binary); if(!fil) { cout<<"File not Found"; exit(0);

} else { fil.read((char\*)&fileobj, sizeof(fileobj)); while(!fil.eof()) { fileobj.show(); cout<<"Press Any Key....For Next Record"<<endl;

//getch() getchar(); fil.read((char\*)&fileobj, sizeof(fileobj)); } } fil.close();

}

void DisplayP() //Function to Display particular Record from Data

File

{ int n; cout<<"Enter Roll no of student that should be searched:"; cin>>n; fil.open("binary.DAT",ios::in| ios::binary); if(!fil) { cout<<"File not Found"; exit(0);

} else { fil.read((char\*)&fileobj, sizeof(fileobj)); while(!fil.eof()) { if(n==fileobj.getn()) { fileobj.show(); cout<<"Press Any Key...."<<endl; getchar();//getch();

} else { cout<<"Press Any Key....For Search"<<endl;

//getch(); getchar();;

} fil.read((char\*)&fileobj, sizeof(fileobj)); } } fil.close();

}

void Modify() //Function to Modify Particular Record from Data

File

{ int n; cout<<"EnterRoll no of student that should be searched:"; cin>>n; fil.open("binary.dat",ios::in| ios::out|ios::binary); if(!fil) { cout<<"File not Found"; exit(0);

} else { fil.read((char\*)&fileobj, sizeof(fileobj)); while(!fil.eof()) { if(n==fileobj.getn()) { fil.seekg(0,ios::cur); cout<<"Enter New Record.."<<endl; fileobj.input();

//fil.seekp(fil.tellg() - sizeof(fileobj)); fil.write((char\*)&fileobj, sizeof(fileobj));

} else { cout<<"Press Any Key....For Search"<<endl;

// getch(); getchar();

} fil.read((char\*)&fileobj, sizeof(fileobj)); } } fil.close();

}

void Delete() //Function to Delete Particular Record from Data

File

{

int n; cout<<"Enter rollno of student that should be Deleted :"; cin>>n; ofstream o;

o.open("new.DAT",ios::out|ios::binary); fil.open("binary.DAT",ios::in| ios::binary); if(!fil) { cout<<"File not Found"; exit(0);

} else

{ fil.read((char\*)&fileobj, sizeof(fileobj)); while(!fil.eof()) { if(n!=fileobj.getn())

{

o.write((char\*)&fileobj, sizeof(fileobj)); } else { cout<<"Press Any Key....For Search"<<endl;

//getch(); getchar();

} fil.read((char\*)&fileobj, sizeof(fileobj)); } }

o.close(); fil.close(); remove("binary.DAT"); rename("new.DAT", "binary.DAT");

}

**Output:**

prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ g++ DSA11.cpp prathmesh1505@prathmesh1505-virtual-machine:~/DSA$ ./a.out

1.Create Data File

2.Add New Record in Data File

3.Display Record From Data File

4.Display Particular Record From Data File

5.Modify Paricular Record From Data File

6.Delete Particular Record From Data File 7.Exit From the Program Enter your Option :

1

Enter the rollno,name,Div and address :45

Prathmesh

A

Hostel

Want to Continue.....n

Display Main Menu

1.Create Data File

2.Add New Record in Data File

3.Display Record From Data File

4.Display Particular Record From Data File

5.Modify Paricular Record From Data File

6.Delete Particular Record From Data File 7.Exit From the Program Enter your Option :

3

\*\*\*\*\*Student Information\*\*\*\*\*\*\*\*\*\*\*\*\*

RollNoName Div Address

45 Prathmesh A Hostel

Press Any Key....For Next Record Display Main Menu a

1.Create Data File

2.Add New Record in Data File

3.Display Record From Data File

4.Display Particular Record From Data File

5.Modify Paricular Record From Data File

6.Delete Particular Record From Data File 7.Exit From the Program Enter your Option :

6

Enter rollno of student that should be Deleted :45

Press Any Key....For Search Display Main Menu a

1.Create Data File

2.Add New Record in Data File

3.Display Record From Data File

4.Display Particular Record From Data File

5.Modify Paricular Record From Data File

6.Delete Particular Record From Data File 7.Exit From the Program Enter your Option : 7 prathmesh1505@prathmesh1505-virtual-machine:~/DSA$