Practical No.1

*C++ Program To read details of 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. */

```
# include <iostream>
# include <cstdlib>
# include <string.h>
using namespace std;
/* Node Declaration */
struct node
{
    char label[10];
    int ch_count;
    struct node *child[10];
}*root;
/*Class Declaration */
class BST
{
    public:
        void create_tree();
void display(node * rl);

        BST()
        {
            root = NULL;
        }
};
void BST::create_tree()
{
int tbooks,tchapters,i,j,k;
```

```
root = new node();
cout<<"Enter name of book";
cin>>root->label;
cout<<"Enter no. of chapters in book";
cin>>tchapters;
root->ch_count = tchapters;
for(i=0;i<tchapters;i++)
{
root->child[i] = new node;
cout<<"Enter Chapter name\n";
cin>>root->child[i]->label;
cout<<"Enter no. of sections in  Chapter: "<<root->child[i]->label;
cin>>root->child[i]->ch_count;
for(j=0;j<root->child[i]->ch_count;j++)
{
root->child[i]->child[j] = new node;
cout<<"Enter Section "<<j+1<<"name\n";
cin>>root->child[i]->child[j]->label;
//cout<<"Enter no. of subsections in "<<rl->child[i]->child[j]->label;
//cin>>rl->child[i]->ch_count;
}
}
}

void BST::display(node * rl)
{
int i,j,k,tchapters;
if(rl != NULL)
{
cout<<"\n-----Book Hierarchy---";
cout<<"\n Book title : "<<rl->label;
tchapters = rl->ch_count;
```

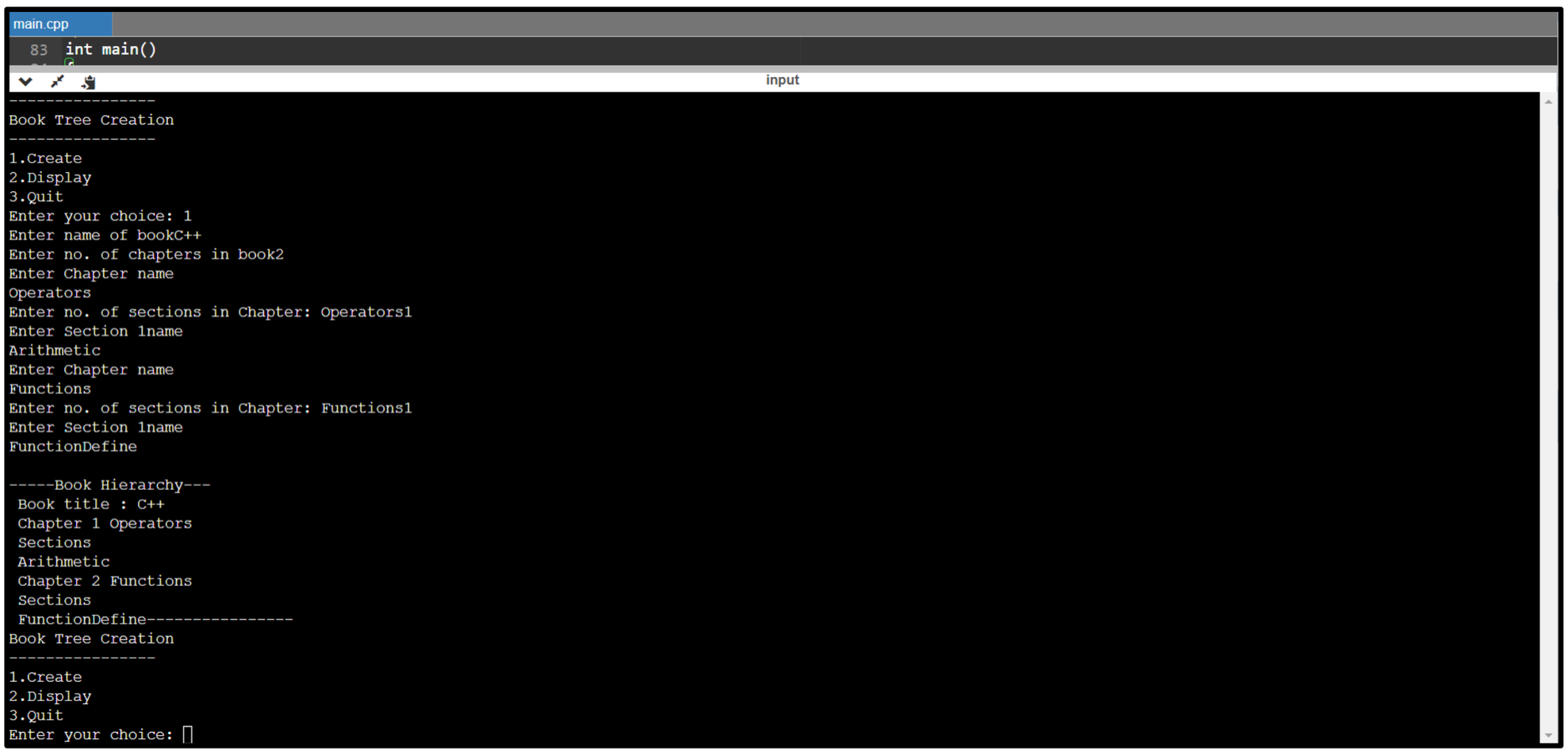
```
for(i=0;i<tchapters;i++)
{
cout<<"\n  Chapter "<<i+1;
cout<<" "<<rl->child[i]->label;
  cout<<"\n Sections";
for(j=0;j<rl->child[i]->ch_count;j++)
{
//cin>>rl->child[i]->child[j]->label;
cout<<"\n "<<rl->child[i]->child[j]->label;

}
}
}
}

/* Main Contains Menu */
int main()
{
    int choice;
    BST bst;
    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:
      bst.create_tree();
case 2:
      bst.display(root);
    break;
case 3:
    exit(1);
        default:
            cout<<"Wrong choice"<<endl;
        }
    }
}
```

Output:



Practical No.2:

```
/*Construct an expression tree from the given prefix and traverse it using
post order traversal 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 stackl
{
node *data[30];
int top;
public:
stackl()
{
top = -l;
}
```

```
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);
```

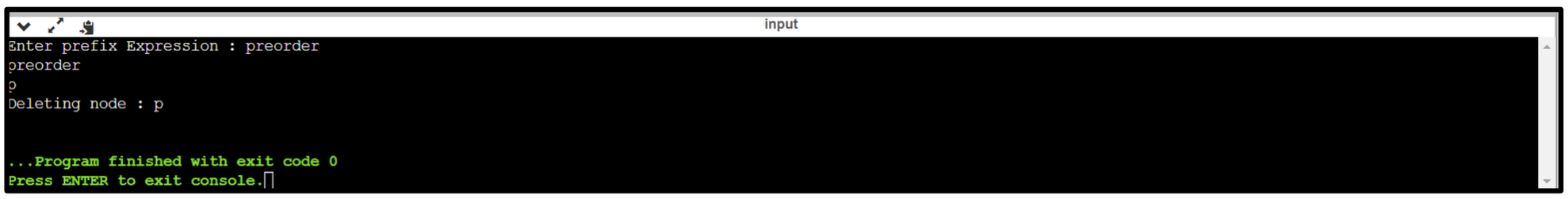
```
}
else if (prefix[i] == '+' || prefix[i] == '*' || prefix[i] == '-' || prefix[i]
== '/')
{
t2 = s.pop();
tl = s.pop();
top->data = prefix[i];
top->left = t2;
top->right = tl;
s.push(top);
}
}
top = s.pop();
}
void tree::display(node *root)
{
if (root != NULL)
{
cout << root->data;
display(root->left);

display(root->right);
}
}
void tree::non_rec_postorder(node *top)
{
stackl sl, s2; /*stack sl is being used for flag . A NULL data implies that the
right subtree has not been visited */
node *T = top;
cout << "\n";
sl.push(T);
while (!sl.empty())
```

```
{
T = sl.pop();
s2.push(T);
if (T->left != NULL)
sl.push(T->left);
if (T->right != NULL)
sl.push(T->right);
}
while (!s2.empty())
{
top = s2.pop();
cout << top->data;
}
}
void tree::del(node *node)
{
if (node == NULL)
return;
/* first delete both subtrees */
del(node->left);
del(node->right);
/* then delete the node */
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.display(t.top);
//cout<<endl;
t.non_rec_postorder(t.top);
t.del(t.top);
// t.display(t.top);
}
```

Output:



Practical No.3:

```
/*Construct an expression tree from the given prefix and traverse it using
post order traversal 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 stackl
{
node *data[30];
int top;
public:
stackl()
{
top = -l;
}
```

```
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();
tl = s.pop();
top->data = prefix[i];
top->left = t2;
top->right = tl;
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)
{
stackl sl, s2; /*stack sl is being used for flag . A NULL data implies that the
right subtree has not been visited */
node *T = top;
cout << "\n";
sl.push(T);
while (!sl.empty())
```

```
{
T = sl.pop();
s2.push(T);
if (T->left != NULL)
sl.push(T->left);
if (T->right != NULL)
sl.push(T->right);
}
while (!s2.empty())
{
top = s2.pop();
cout << top->data;
}
}
void tree::del(node *node)
{
if (node == NULL)
return;
/* first delete both subtrees */
del(node->left);
del(node->right);
/* then delete the node */
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.display(t.top);
//cout<<endl;
t.non_rec_postorder(t.top);
t.del(t.top);
// t.display(t.top);
}
```

Output:

Practical No.4:

/*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 the airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph.

*/

```
#include <iostream>
#include <queue>
using namespace std;
int adj_mat[50][50] = {0, 0};
int visited[50] = {0};
void dfs(int s, int n, string arr[])
{
    visited[s] = 1;
    cout << arr[s] << " ";
    for (int i = 0; i < n; i++)
    {
        if (adj_mat[s][i] && !visited[i])
            dfs(i, n, arr);
    }
}
void bfs(int s, int n, string arr[])
{
    bool visited[n];
    for (int i = 0; i < n; i++)
        visited[i] = false;
int v;
queue<int> bfsq;
if (!visited[s])
{
```

```
cout << arr[s] << " ";
bfsq.push(s);
visited[s] = true;
while (!bfsq.empty())
{
    v = bfsq.front();
    for (int i = 0; i < n; i++)
    {
        if (adj_mat[v][i] && !visited[i])
        {
            cout << arr[i] << " ";
            visited[i] = true;
            bfsq.push(i);
        }
    }
    bfsq.pop();
}
}
int main()
{
    cout << "Enter no. of cities: ";
    int n, u;
    cin >> n;
    string cities[n];
    for (int i = 0; i < n; i++)
    {
    cout << "Enter city #" << i << " (Airport Code): ";
    cin >> cities[i];
    }

cout << "\nYour cities are: " << endl;
```

```
for (int i = 0; i < n; i++)
    cout << "city #" << i << ": " << cities[i] << endl;
for (int i = 0; i < n; i++)
{
    for (int j = i + 1; j < n; j++)
    {
        cout << "Enter distance between " << cities[i] << " and " << cities[j] << " : ";
        cin >> adj_mat[i][j];
        adj_mat[j][i] = adj_mat[i][j];
    }
}
cout << endl;
for (int i = 0; i < n; i++)
    cout << "\t" << cities[i] << "\t";
for (int i = 0; i < n; i++)
{
    cout << "\n"
         << cities[i];
    for (int j = 0; j < n; j++)
        cout << "\t" << adj_mat[i][j] << "\t";
    cout << endl;
}
cout << "Enter Starting Vertex: ";
cin >> u;
cout << "DFS: ";
dfs(u, n, cities);
    cout << endl;
    cout << "BFS: ";
    bfs(u, n, cities);
    return 0;
}
```

Output:

input

```
Enter no. of cities: 2
Enter city #0 (Airport Code): 1001
Enter city #1 (Airport Code): 1005

Your cities are:
city #0: 1001
city #1: 1005
Enter distance between 1001 and 1005 : 20

        1001            1005
1001    0               20

1005    20              0
Enter Starting Vertex: 1
DFS: 1005 1001
BFS: 1005 1001

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

Practical No.5:

```
#include <iostream>
using namespace std;
int main()
{
int size;
cout<<"enter the size of hash table"<<endl;
cin>>size;
int arr[size],arl[size];
int key,L,no_of_elements,pre;
for(int i=0;i<size;i++)
{
arr[i]=0;
arl[i]=-l;
}
for(int i=0;i<size;i++)
{
cout<<i<<"\t"<<arr[i]<<"\t"<<arl[i];
cout<<endl;
}
cout<<"how many elemnts want to store";
cin>>no_of_elements;
for(int i=0;i<no_of_elements;i++)
{
cout<<"enter key";
cin>>key;

int location=key%size;
int pre=location;
if(arr[location]==0)
{
```

```
arr[location]=key;
}
else
{
while(arr[location]!=0)
{
location++;
}
//cout<<"value of location"<<location;
arr[location]=key;
if(arl[pre]!=-l)
{
int s=arl[pre];
arl[s]=location;
}
else
{
{
arl[pre]=location;
}
}
for(int i=0;i<size;i++)

{
cout<<i<<"\t"<<arr[i]<<"\t"<<arl[i];
cout<<endl;
}
}
return 0;
}
```

Output:


input
enter the size of hash table
4
0 0 -1
1 0 -1
2 0 -1
3 0 -1
how many elemnts want to store2
enter key10
0 0 -1
1 0 -1
2 10 -1
3 0 -1
enter key12
0 12 -1
1 0 -1
2 10 -1
3 0 -1
...Program finished with exit code 0
Press ENTER to exit console.

Practical No.6:

```
#include<iostream>
#include<cstdlib>
#include<string>
#include<cstdio>
using namespace std;
const int T_S = 200;
class HashTableEntry {
public:
int k;
int v;
HashTableEntry(int k, int v) {
this->k= k;
this->v = v;
}
};
class HashMapTable {
private:
HashTableEntry **t;
public:
HashMapTable() {
t = new HashTableEntry * [T_S];
for (int i = 0; i< T_S; i++) {
t[i] = NULL;
}
}

int HashFunc(int k) {
return k % T_S;
}
void Insert(int k, int v) {
```

```
int h = HashFunc(k);
while (t[h] != NULL &amp;amp;&amp;amp; t[h]->k != k) {
h = HashFunc(h + 1);
}
if (t[h] != NULL)
delete t[h];
t[h] = new HashTableEntry(k, v);
}
int SearchKey(int k) {
int h = HashFunc(k);
while (t[h] != NULL &amp;amp;&amp;amp; t[h]->k != k) {
h = HashFunc(h + 1);
}
if (t[h] == NULL)
return -1;
else
return t[h]->v;
}
void Remove(int k) {
int h = HashFunc(k);
while (t[h] != NULL) {

if (t[h]->k == k)
break;
h = HashFunc(h + 1);
}
if (t[h] == NULL) {
cout<<"No Element found at key "<<k<<endl;
return;
} else {
delete t[h];
}
```

```
cout<<"Element Deleted"<<endl;
}
~HashMapTable() {
for (int i = 0; i < T_S; i++) {
if (t[i] != NULL)
delete t[i];
delete[] t;
}
}
};
int main() {
HashMapTable hash;
int k, v;
int c;
while (1) {

cout<<"1.Insert element into the table"<<endl;
cout<<"2.Search element from the key"<<endl;
cout<<"3.Delete element at a key"<<endl;
cout<<"4.Exit"<<endl;
cout<<"Enter your choice: ";
cin>>c;
switch(c) {
case 1:
cout<<"Enter element to be inserted: ";
cin>>v;
cout<<"Enter key at which element to be inserted: ";
cin>>k;
hash.Insert(k, v);
break;
case 2:
cout<<"Enter key of the element to be searched: ";
```

```
cin>>k;
if (hash.SearchKey(k) == -1) {
cout<<"No element found at key "<<k<<endl;
continue;
} else {
cout<<"Element at key "<<k<<" : ";
cout<<hash.SearchKey(k)<<endl;
}
break;

case 3:
cout<<"Enter key of the element to be deleted: ";
cin>>k;
hash.Remove(k);
break;
case 4:
exit(1);
default:
cout<<"\nEnter correct option\n";
}
}
return 0;
}
```

Output:

Practical No.7:

```
#include <iostream>
using namespace std;
int main()
{
int size,location,key;
cout<<"Enter size of hash table: "<<endl;
cin>>size;
int arr[size],arrl[size];
location= key%size ;
//To create 1st index coloumn...
for(int i=0;i<size;i++)
{
arr[i]=i;
}
//To create 2nd data coloumn...
for(int i=0;i<size;i++)
{
arrl[i]=0;
}
//Print the hash table...
cout<<"\n";
for(int i=0;i<size;i++){
cout<<arr[i]<<" " <<arrl[i]<<endl;
}
//Take input from user...
int no_of_elements;
cout<<"Enter how many elements you want to store: "<<endl;
cin>>no_of_elements;
for(int i=0;i<no_of_elements;i++)
{
```

```
cout<<"Enter element: "<<endl;
cin>>key;
location= key%size ;
if(arr[location]==0)
{
arr[location]=key;
}
else
{
location++;
arr[location]=key;
}
}
//Printing final hash table...
cout<<"\n";
for(int i=0;i<size;i++){
cout<<arr[i]<<" " <<arr[i]<<endl;
}
return 0;
}
```

Output:

Practical No.8:

/*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. Make use of appropriate data structures.

(Using BST)

*/

```
#include <iostream>
#include <string.h>
#define MAX 10
using namespace std;
class node
{
private:
char keyword[MAX];
char meaning[MAX];
node *left;
node *right;
public:
node();
node(char [],char []);
friend class BST;
};
node::node()
{
keyword[0] = '\0';
meaning[0] = '\0';
left = NULL;
right = NULL;
}
node::node(char key[],char mean[])
```

```
{
strcpy(keyword,key);
strcpy(meaning,mean);
left = NULL;
right = NULL;
}
class BST
{
private:
node *root;
void inorder(node *root);
void inorderrev(node *root);
node * insert(node * root,node *);
node* search(char key[]);

node * remove(node *root,char key[]);
public:
BST();
void update(char []);
void recInsert(char [] , char[]);
node *search(node *, char []);
void find(char key[]);
void printAscending();
void printDescending();
void removeword(char val[]);
};
BST::BST()
{
root = NULL;
}
void BST::recInsert(char k[], char m[])
{
```

```
node *newnd=new node(k,m);
root = insert(root,newnd);
}
node * BST::insert(node * root,node *newnd)
{
if(root==NULL)
root = newnd;
else if (strcmp(newnd->keyword, root->keyword)<0)
root->left=insert(root->left,newnd);
else if
(strcmp(newnd->keyword, root->keyword)>0)
root->right=insert(root->right,newnd);
else
cout<<"\nDuplicate value";
return root;
}
node * BST::remove(node *root,char key[])
{
if(root==NULL)
return NULL;
else if (strcmp(key,root->keyword)<0)
root->left=remove(root->left,key);
else if (strcmp(key,root->keyword)>0)
root->right=remove(root->right,key);

else
{
if(root->right!=NULL)
{
node *in_succ = root->right;
while(in_succ->left!=NULL)
{
```

```
in_succ = in_succ->left;
}
strcpy(root->keyword,in_succ->keyword);
strcpy(root->meaning,in_succ->meaning);
root->right=remove(root->right,in_succ->keyword);
}
else
return root->left;
}
return root;
}
void BST::removeword(char val[])
{
remove(root,val);
}
void BST::update(char k[])
{
node *tempnd;
tempnd = search(k);
if(tempnd == NULL)
cout<<"\nWord not present for Update:";
else
{
cout<<"\nEnter new meaning for this word";
cin>>tempnd->meaning;
}
}
node *BST::search(char val[])
{
node *tempnd=root;
tempnd=search(tempnd,val);
```

```
return tempnd;
}
node *BST::search(node *tempnd, char k[])
{
if(tempnd!= NULL)
{
if(strcmp(k,tempnd->keyword)==0)
return tempnd;
else if(strcmp(k, tempnd->keyword)<0)
search(tempnd->left, k);
else
search(tempnd->right, k);
}
//
if
else

return NULL;


//not found
}
void BST::find(char val[])
{
node *tempnd;
tempnd=search(val);
if (tempnd == NULL)
{
cout<<endl<<"Not found\n";
}
else
{
cout<<endl<<"Found...";
```

```
cout<<"\nMeaning is "<<tempnd->meaning;
}
}
//find
void BST::inorder(node *root)
{
if(root!= NULL)
{
inorder(root->left);
cout<<"\n"<<root->keyword<<":"<<root->meaning;
inorder(root->right);
}
}
void BST::inorderrev(node *root)
{
if(root!= NULL)
{
inorderrev(root->right);
cout<<"\n"<<root->keyword<<":"<<root->meaning;
inorderrev(root->left);
}
}
void BST::printAscending()
{
inorder(root);
}
void BST::printDescending()
{
inorderrev(root);
}
int menu()
{
```

```
int choice;
cout<<"\nDICTIONARY APPLICATION";
cout<<"\n\tl. Insert ";
cout<<"\n\t2. Update ";
cout<<"\n\t3. Delete ";
cout<<"\n\t4. Print Ascending ";
cout<<"\n\t5. Print Descending";
cout<<"\n\t6. Find ";
cout<<"\n\t7. Exit ";
cout<<"\nEnter your Choice \t";
cin>>choice;
return choice;
}
//menu
int main()
{
BST t;
char k[MAX],m[MAX];
char keyword[40];
int n,choice;
while(l)
{
choice=menu();;
switch(choice)
{
case l: cout<<"\nHow many values to insert";
cin>>n;
for(int i=0;i<n;i++)
{
cout<<"\nEnter new Keyword :";
cin>>k;
cout<<"\nEnter meaning of "<<k<<" : ";
```

```
cin>>m;
t.recInsert(k,m);
}
break;
case 2: cout<<"\nEnter keyword to be found:";
cin>>k;
t.update(k);
break;
case 3: cout<<"Enter the keyword :";
cin>>keyword;
t.removeword(keyword);
break;
case 4: cout<<"\nPrint Dictionary in Ascending Order: ";
t.printAscending();
break;
case 5: cout<<"\nPrint Dictionary in Descending Order: ";
t.printDescending();
break;
case 6: cout<<"Enter the keyword :";
cin>>keyword;
t.find(keyword);
break;
case 7: cout <<"\nProgram ending....\n";
return 0;
default: cout <<"\nEnter correct choice...\n";
}
//switch
}//
while
return 0;
}
//main
```

Output:

```
DICTIONARY APPLICATION
        1. Insert
        2. Update
        3. Delete
        4. Print Ascending
        5. Print Descending
        6. Find
        7. Exit
Enter your Choice       1

How many values to insert2

Enter new Keyword :FYI

Enter meaning ofFYI:For Your Information

Enter new Keyword :
Enter meaning ofYour:
DICTIONARY APPLICATION
        1. Insert
        2. Update
        3. Delete
        4. Print Ascending
        5. Print Descending
        6. Find
        7. Exit
Enter your Choice
```

Practical No.9:

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

#include<iostream>
using namespace std;
class Heap
{
int n;
int *minheap,*maxheap;
public:
void get();
void displayMin(){cout<<"Minimum marks are :"<<minheap[0]<<endl;}
void displayMax(){cout<<"Maximum marks are :"<<maxheap[0]<<endl;}
void upAdjust(bool,int);
};
void Heap::get()
{
cout<<"Enter number of students."<<endl;
cin>>n;
int k;
minheap=new int[n];
maxheap=new int[n];
cout<<"Enter marks of students."<<endl;
for(int i=0;i<n;i++)
{
cin>>k;
minheap[i]=k;
upAdjust(0,i);
maxheap[i]=k;
upAdjust(1,i);
}
```

```
}
void Heap::upAdjust(bool m,int l)
{
int s;

if(!m)
{
while(minheap[(l-1)/2]<minheap[l])
{
s=minheap[l];
minheap[l]=minheap[(l-1)/2];
minheap[(l-1)/2]=s;
l=(l-1)/2;
if(l==-1)
break;

}
}
else
{
while(maxheap[(l-1)/2]>maxheap[l])
{
s=maxheap[l];
maxheap[l]=maxheap[(l-1)/2];
maxheap[(l-1)/2]=s;
l=(l-1)/2;
if(l==-1)
break;

}
}
}
```

```
main()
{
Heap H;
H.get();
H.displayMin();
H.displayMax();
return(0);
}
```

Output:

```
Enter number of students:
4
Enter marks of students:
70
90
68
89
Minimum marks are :90
Maximum marks are :90


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

Practical No.10:

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

```
#include<iostream>
#include<fstream>
#include<cstring>
using namespace std;
class tel
 {
 public:
int rollNo,roll1;
char name[10];
char div;
char address[20];
void accept()
{
cout<<"\n\tEnter Roll Number : ";
cin>>rollNo;
cout<<"\n\tEnter the Name : ";
cin>>name;
cout<<"\n\tEnter the Division:";
cin>>div;
cout<<"\n\tEnter the Address:";
cin>>address;
}
        void accept2()
        {
                cout<<"\n\tEnter the Roll No. to modify : ";
                cin>>rollNo;
```

```
}
void accept3()
{
      cout<<"\n\tEnter the name to modify : ";
      cin>>name;
}
int getRollNo()
      {
       return rollNo;
      }
 void show()
 {
cout<<"\n\t"<<rollNo<<"\t\t"<<name<<"\t\t"<<div<<"\t\t"<<address;
 }

};
int main()
{
int
i,n,ch,chl,rec,start,count,add,nl,add2,start2,n2,y,a,b,on,oname,add3,start3,n3,yl,add4,start4,n4;
char name[20],name2[20];
tel tl;
count=0;
fstream g,f;
do
{
cout<<"\n>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<";
cout<<"\nl.Insert and overwrite\n2.Show\n3.Search &amp; Edit(number)\n4.Search &amp;
Edit(name)\n5.Search
&amp; Edit(onlynumber)\n6.Search &amp; edit(only name)\n 7.Delete a Student Record\n 8.Exit\n\tEnter the
Choice\t:";
cin>>ch;
```

```
switch(ch)
{
case 1:
f.open("StuRecord.txt",ios::out);
x:tl.accept();
f.write((char*) &amp;tl,(sizeof(tl)));
cout<<"\nDo you want to enter more records?\n1.Yes\n2.No";
cin>>ch1;
if(ch1==1)
goto x;
else
{
f.close();
break;
}
case 2:
f.open("StuRecord.txt",ios::in);
f.read((char*) &amp;tl,(sizeof(tl)));
//cout<<"\n\tRoll No.\t\tName \t\t Division \t\t Address";
while(f)
{
tl.show();
f.read((char*) &amp;tl,(sizeof(tl)));
}
f.close();
break;
case 3:
cout<<"\nEnter the roll number you want to find";
cin>>rec;
f.open("StuRecord.txt",ios::in|ios::out);
f.read((char*)&amp;tl,(sizeof(tl)));
while(f)
```

```
{
if(rec==tl.rollNo)
{
cout<<"\nRecord found";
add=f.tellg();
f.seekg(0,ios::beg);

      start=f.tellg();
nl=(add-start)/(sizeof(tl));
f.seekp((nl-l)*sizeof(tl),ios::beg);
tl.accept();
f.write((char*) &amp;tl,(sizeof(tl)));
f.close();
count++;
break;
}
f.read((char*)&amp;tl,(sizeof(tl)));
    }
if(count==0)
       cout<<"\nRecord not found";
f.close();
break;
case 4:
 cout<<"\nEnter the name you want to find and edit";
 cin>>name;
f.open("StuRecord.txt",ios::in|ios::out);
f.read((char*)&amp;tl,(sizeof(tl)));
while(f)
{
 y=(strcmp(name,tl.name));
 if(y==0)
 {
```

```
cout<<"\nName found";
add2=f.tellg();
f.seekg(0,ios::beg);
start2=f.tellg();
n2=(add2-start2)/(sizeof(tl));
f.seekp((n2-1)*sizeof(tl),ios::beg);
tl.accept();
f.write((char*) &amp;tl,(sizeof(tl)));
f.close();
break;
}
        f.read((char*)&amp;tl,(sizeof(tl)));
}
break;
    case 5:
        cout<<"\n\tEnter the roll number you want to modify";
        cin>>on;
        f.open("StuRecord.txt",ios::in|ios::out);
        f.read((char*) &amp;tl,(sizeof(tl)));
        while(f)
        {
         if(on==tl.rollNo)
         {
           cout<<"\n\tNumber found";
           add3=f.tellg();
           f.seekg(0,ios::beg);
           start3=f.tellg();
           n3=(add3-start3)/(sizeof(tl));
           f.seekp((n3-1)*(sizeof(tl)),ios::beg);

           tl.accept2();
           f.write((char*)&amp;tl,(sizeof(tl)));
```

```
            f.close();
            break;
          }
          f.read((char*)&amp;tl,(sizeof(tl)));
        }
        break;
    case 6:
          cout<<"\nEnter the name you want to find and edit";
 cin>>name2;
f.open("StuRecord.txt",ios::in|ios::out);
f.read((char*)&amp;tl,(sizeof(tl)));
while(f)
{
 yl=(strcmp(name2,tl.name));
 if(yl==0)
 {
 cout<<"\nName found";
 add4=f.tellg();
 f.seekg(0,ios::beg);
 start4=f.tellg();
 n4=(add4-start4)/(sizeof(tl));
 f.seekp((n4-l)*sizeof(tl),ios::beg);
 tl.accept3();
 f.write((char*) &amp;tl,(sizeof(tl)));
 f.close();
 break;
 }
        f.read((char*)&amp;tl,(sizeof(tl)));
}
break;
   case 7:
    int roll;
```

```
 cout<<"Please Enter the Roll No. of Student Whose Info You Want to Delete: ";
cin>>roll;
f.open("StuRecord.txt",ios::in);
g.open("temp.txt",ios::out);
f.read((char *)&amp;tl,sizeof(tl));
while(!f.eof())
{
  if (tl.getRollNo() != roll)
    g.write((char *)&amp;tl,sizeof(tl));
   f.read((char *)&amp;tl,sizeof(tl));
}
cout << "The record with the roll no. " << roll << " has been deleted " << endl;
 f.close();
 g.close();
 remove("StuRecord.txt");
 rename("temp.txt","StuRecord.txt");
  break;
   case 8:
      cout<<"\n\tThank you";
      break;

       }
  }while(ch!=8);
}
```

Output:

input

```
>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search and Edit(number)
4.Search and Edit(name)
5.Search and Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :1

        Enter Roll Number : 2

        Enter the Name : Anushka

        Enter the Division:A

        Enter the Address:Pune

Do you want to enter more records?
1.Yes
2.No2

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search and Edit(number)
4.Search and Edit(name)
5.Search and Edit(onlynumber)
```

Practical No.II:

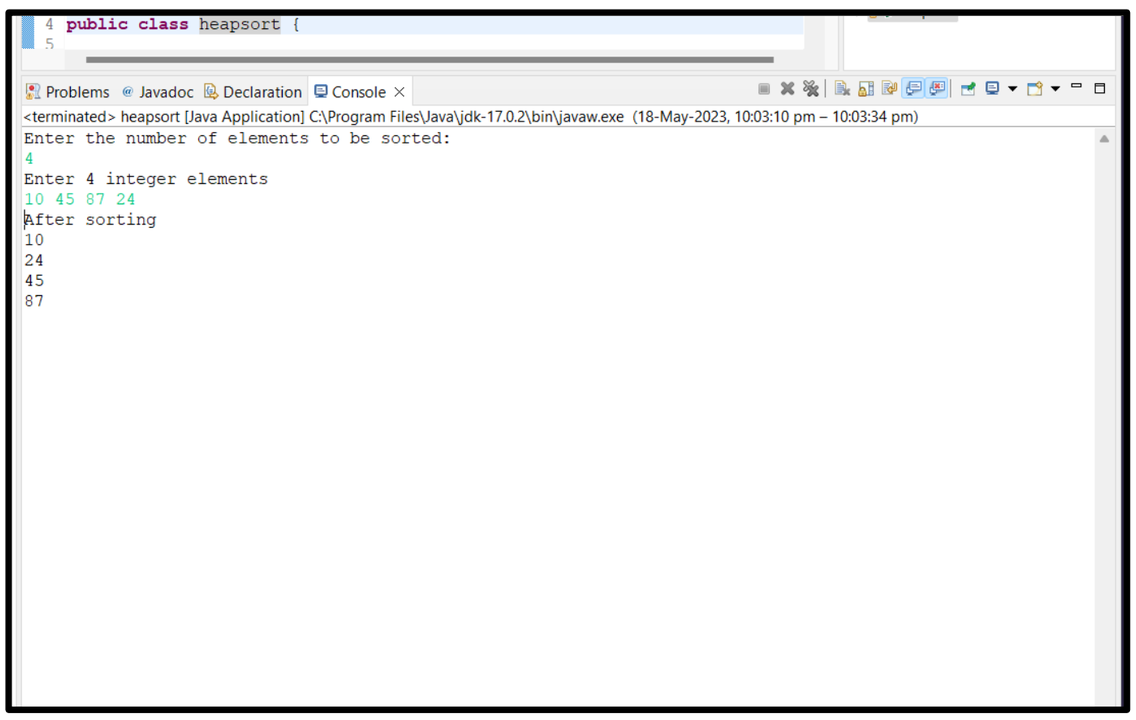
Implement the Heap/Shell sort algorithm implemented in Java demonstrating heap/shell

data structure with modularity of programming language.

```
import java.util.*;
public class vivek_al2 {
private static int N;
    public static void sort(int arr[]){
heapMethod(arr);
        for (int i = N; i > 0; i--){
            swap(arr,0, i);
            N = N-1;
            heap(arr, 0);
        }
    }
    public static void heapMethod(int arr[]){
        N = arr.length-1;
        for (int i = N/2; i >= 0; i--)
            heap(arr, i);
    }
    public static void heap(int arr[], int i){
        int left = 2*i ;
        int right = 2*i + 1;
        int max = i;
        if (left <= N &amp;&amp; arr[left] > arr[i])
            max = left;
if (right <= N &amp;&amp; arr[right] > arr[max])
            max = right;
        if (max != i){
            swap(arr, i, max);
            heap(arr, max);
        } }
    public static void swap(int arr[], int i, int j){
```

```
    int tmp = arr[i];
    arr[i] = arr[j];
    arr[j] = tmp;
}
public static void main(String[] args) {
    Scanner in = new Scanner( System.in );
    int n;
    System.out.println("Enter the number of elements to be sorted:");
    n = in.nextInt();
    int arr[] = new int[ n ];
    System.out.println("Enter "+ n +" integer elements");
    for (int i = 0; i < n; i++)
        arr[i] = in.nextInt();
    sort(arr);
    System.out.println("After sorting ");
    for (int i = 0; i < n; i++)
        System.out.println(arr[i]+" ");
        System.out.println();
    }
}
```

Output:



Practical No.12:

Write a Java program which will demonstrate a concept of

Interfaces and packages: In this assignment design and

use of customized interfaces and packages for a specific

application are expected.

```
// This program has two codes one is normal code and another is package code
//-------------------------------------lst code ----------------------------------------
import java.util.*;
import sum.*;
interface Area{
  final static float pi=3.14F;
  float compute(float x,float y);
}
class rectangle implements Area{
  public  float compute (float x, float y)
  {
    return(x*y);
  }
}
class circle implements Area
{
 public float compute(float x , float y)
 {
   return(pi*x*x);
 }
}

class interpack
{
public static void main (String args[])
 {
  rectangle rect = new rectangle();
```

```
circle cir = new circle();
pack p=new pack();
p.add(10,20);
Area X;
X=rect;
System.out.println("Area of rectangle:" + X.compute(10,20));
X= cir;
System.out.println("Area of circle:" + X.compute(10,0));
 }
}
//----------------------2nd code-----------------------------------------------
package sum;
public class pack
 {
  public  void add(double x,double y)
  {
   double z;
   z=x+y;
   System.out.println(" net payment  = "+z);
   }

}
```

Output: