/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 1

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

'''

## Problem Statement:

**Implement all the functions of a dictionary (ADT) using Seperate Chaining.**

## 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).

============================================================================

*Name : Assignment\_2.py Author : Sahil Deshmukh*

============================================================================ '''

class SeparateHash: def init (self,n):

self.n = n

self.v = [[] for i in range(n)]

def getHashIndex(self, x): return x % self.n

def add(self, x):

i = self.getHashIndex(x) if x not in self.v[i]:

self.v[i].append(x)

def del\_(self, x):

i = self.getHashIndex(x) if x in self.v[i]:

self.v[i].remove(x)

print(x, "deleted!") else:

print("No Element Found!")

def displayHash(self): for i in range(self.n):

print(i, end=" -> ")

if(len(self.v[i])==0):

print("NULL",end=" ") for j in self.v[i]:

print(j, end=" ") print()

ch = 'y'

obj = SeparateHash(10) while(ch=='y' or ch=='Y'):

n = int(input("\n1.Insert Element\n2.Find Key\n3.Display Element.\n4.Delete Element\nEnter your choice: "))

if(n==1):

num = int(input("How many elements you want to insert?: ")) for i in range(num):

key = int(input("Enter element to insert : ")) obj.add(key)

if(n==2):

key = int(input("Enter element to find : "))

print(" Value of index is ", obj.getHashIndex(key))

if(n==3): obj.displayHash()

if(n==4):

key = int(input("Enter element to delete : ")) obj.del\_(key)

ch = input("Do you want to continue??(y/n): ")

Output :

1.Insert Element 2.Find Key 3.Display Element. 4.Delete Element Enter your choice: 1

How many elements you want to insert?: 5 Enter element to insert : 10

Enter element to insert : 20 Enter element to insert : 30 Enter element to insert : 40 Enter element to insert : 50

Do you want to continue??(y/n): Y

1.Insert Element 2.Find Key 3.Display Element. 4.Delete Element Enter your choice: 2

Enter element to find : 10 Value of index is 0

Do you want to continue??(y/n): Y

1.Insert Element 2.Find Key 3.Display Element. 4.Delete Element Enter your choice: 3 0 -> 10 20 30 40 50

1. -> NULL
2. -> NULL
3. -> NULL
4. -> NULL
5. -> NULL
6. -> NULL
7. -> NULL
8. -> NULL
9. -> NULL

Do you want to continue??(y/n): Y

1.Insert Element 2.Find Key 3.Display Element. 4.Delete Element Enter your choice: 4

Enter element to delete : 30 30 deleted!

Do you want to continue??(y/n): N

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

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 2

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

Problem Statement:

To create ADT that implements the "set" concept. a. Add (new Element) -Place a value into the set ,

b. Remove (element) Remove the value c. Contains (element) Return true if element is in collection,

d. Size () Return number of values in collection Iterator () Return an iterator used to loop over collection, e. Intersection of two sets , f. Union of two sets, g. Difference between two sets, h. Subset

======================================================================

*Name:- Assignment\_2.py Author:- Sahil Deshmukh*

=====================================================================\*/

setA=set() setB=set()

n=int(input("enter the limit of setA:")) m=int(input("enter the limit of setB:")) def add():

for i in range(n):

ele=input("enter element for setA:") setA.add(ele)

print(setA); def add1():

for i in range(m):

ele=input("enter element for setB:") setB.add(ele)

print(setB); def remove():

ele=input("enter element to remove:") setA.remove(ele)

print(setA,"\n"); def contain():

contains = 7 in setA print("7 present is setA:"); print(contains);

def union():

print("union of sets:"); print(setA|setB,"\n");

def intersection(): print("Intersections:"); print(setA.intersection(setB),"\n");

def difference(): print("difference of sets:"); print(setA-setB,"\n");

def length(): print("Lenth of setA:"); print(len(setA)); print("Lenth of setB:"); print(len(setB));

def subset(): print(setB.issubset(setA)); add()

add1() remove() contain() union() intersection() difference() length() subset()

OUTPUT:-

enter the limit of setA:3 enter the limit of setB:3 enter element for setA:1

{'1'}

enter element for setA:2

{'1', '2'}

enter element for setA:3

{'1', '3', '2'}

enter element for setB:1

{'1'}

enter element for setB:7

{'7', '1'}

enter element for setB:8

{'7', '1', '8'}

enter element to remove:2

{'1', '3'}

7 present is setA: False

union of sets:

{'1', '3', '7', '8'}

Intersections:

{'1'}

difference of sets:

{'3'}

Lenth of setA:

2

Lenth of setB:

3

False

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 3

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**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.

============================================================================

*Name : Assignment\_3.cpp Author : Sahil Deshmukh*

============================================================================\*/

#include<iostream> using namespace std; struct node

{

string data; node\* child[10]; int count;

};

class csbook

{

public:

node\* root; csbook()

{

root=NULL;

}

void create(); void display();

};

void csbook::create()

{

int i,j,k; root=new node;

cout<<"\nEnter the book name : "; cin>>root->data;

cout<<"\nHow many chapters : "; cin>>root->count;

for(i=1;i<=root->count;i++)

{

cout<<"\nEnter the chapter "<<i<<" name : "; root->child[i]=new node;

cin>>root->child[i]->data; cout<<"\nHow many sections : "; cin>>root->child[i]->count; for(j=1;j<=root->child[i]->count;j++)

{

root->child[i]->child[j]=new node; cout<<"\nEnter the section "<<j<<" name : "; cin>>root->child[i]->child[j]->data;

cout<<"\nHow many sub sections : "; cin>>root->child[i]->child[j]->count; for(k=1;k<=root->child[i]->child[j]->count;k++)

{

root->child[i]->child[j]->child[k]=new node; cout<<"\nEnter the sub section "<<k<<" name : "; cin>>root->child[i]->child[j]->child[k]->data;

}

}

}

}

void csbook::display()

{

int i,j,k;

cout<<"the book name : "<<root->data<<"\n"; for(i=1;i<=root->count;i++)

{

cout<<"The chapter "<<i<<" : "<<root->child[i]->data<<"\n"; for(j=1;j<=root->child[i]->count;j++)

{

cout<<"The section of chapter "<<root->child[i]->data<<" : "<<root->child[i]->child[j]->data<<"\n"; for(k=1;k<=root->child[i]->child[j]->count;k++)

{

cout<<"The sub-section of section "<<root->child[i]->child[j]->data<<" : "<<root->child[i]->child[j]->child[k]->data<<"\n";

}

}

}

}

int main()

{

csbook p; char ans; int ch;

do

{

cout<<"-----------MAIN MENU \n";

cout<<"1.Create";

cout<<"\n2.Display \nEnter your choice: "; cin>>ch;

switch(ch)

{

case 1 :

p.create(); break;

case 2 :

p.display(); break;

}

cout<<"\nDo u want to continue??(y/n): "; cin>>ans;

}while(ans=='y'||ans=='Y'); return 0;

}

Output :

MAIN MENU

1.Create 2.Display

Enter your choice: 1

Enter the book name : dsal How many chapters : 2

Enter the chapter 1 name : hashing How many sections : 1

Enter the section 1 name : psg How many sub sections : 1

Enter the sub section 1 name : asg Enter the chapter 2 name : tree How many sections : 1

Enter the section 1 name : bst How many sub sections : 1

Enter the sub section 1 name : bt Do u want to continue[y/n): n

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 4

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**Understand the implementation of the Binary Search Tree like find minimum max,insert, delete,find,inorder,preorder etc.**

============================================================================

*Name : Assignment\_4.cpp Author :Sahil Deshmukh*

============================================================================\*/

#include <iostream> #define MAX 100

using namespace std; class node

{

public:

int data; node \*left; node \*right;

node \*create(node \*root)

{

node \*temp; char ans2; do

{

temp = new node;

temp->left=temp->right=NULL; cout<<"\nEnter the data: "; cin>>temp->data; if(root==NULL)

root=temp;

else

insertnode(root,temp);

cout<<"\n\nDo u want 2 add another node?(y/n) : "; cin>>ans2;

}while(ans2=='y'|| ans2=='Y'); return(root);

}

void insertnode(node \*root,node \*temp)

{

if(temp->data < root->data)

{

if(root->left==NULL)

root->left=temp;

else

{

}

}

insertnode(root->left,temp);

if(temp->data > root->data)

{

if(root->right==NULL)

root->right=temp;

else

}

}

insertnode(root->right,temp);

void printInorder(node\*root)

{

if (root == NULL) return; printInorder(root->left);

cout << root->data << " "; printInorder(root->right);

}

void printPreOrder(node \* root)

{

if (root == NULL) return;

cout << root->data << " "; printPreOrder(root->left); printPreOrder(root->right);

}

node\* findmin(node \*root)

{

node\* current = root;

while (current && current->left != NULL) current = current->left;

return current;

}

node\* findmax(node \*root)

{

node\* current = root;

while (current && current->right != NULL) current = current->right;

return current;

}

node\* deleteNode(struct node\* root, int key)

{

if (root == NULL) return root;

if (key < root->data)

root->left = deleteNode(root->left, key);

else if (key > root->data)

root->right = deleteNode(root->right, key);

else

{

// node has no child

if (root->left == NULL and root->right == NULL) return NULL;

// node with only one child or no child else if (root->left == NULL)

{

node\* temp = root->right; free(root);

return temp;

}

else if (root->right == NULL)

{

node\* temp = root->left; free(root);

return temp;

}

// node with two children: Get the inorder successor (smallest in the right subtree) node\* temp = findmin(root->right);

// Copy the inorder successor's content to this node root->data = temp->data;

// Delete the inorder successor

root->right = deleteNode(root->right, temp->data);

}

return root;

}

node\* find(node\* root, int key)

{

if (root == NULL || root->data == key)

{

return root;

}

if (root->data < key)

{

return find(root->right, key);

}

return find(root->left, key);

}

};

int main()

{

node n1; int ch2; char ans1;

node \*root=NULL; do

{

cout<<"\n1.CREATE TREE\n2.Find Minimum\n3.Find Maximum\n4.Print INORDER\n5.Print PREORDER\n6.Find Key\n7.Deletenode\n8.Exit";

cout<<"\n\nEnter the choice: "; cin>>ch2;

switch(ch2)

{

case 1:

root=n1.create(root); break;

case 2:

cout<<"\nMinimum Element is : "<<(n1.findmin(root))->data;

break;

case 3:

cout<<"\nMaximum Element is : "<<(n1.findmax(root))->data; break;

case 4:

n1.printInorder(root); break;

case 5:

n1.printPreOrder(root); break;

case 6:

int key3; node \*res;

cout<<"\nEnter key to search: ";

cin>>key3;

res = n1.find(root, key3); if(res->data)

cout<<"\nElement found...";

else

cout<<"\nElement Not Found...";

break;

case 7:

int key2; node\* res1;

cout<<"\nEnter key to delete: "; cin>>key2;

res1 = n1.deleteNode(root, key2); if(res1->data)

cout<<"\nDeleted Successfully...";

else

cout<<"\nElement Not Found...";

break;

}

cout<<"\nDo u want to continue...?(y/n) : "; cin>>ans1;

}while(ans1=='y' || ans1=='Y'); return 0;

}

**Output**

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 1 Enter the data: 10

Do u want 2 add another node?(y/n) : y Enter the data: 20

Do u want 2 add another node?(y/n) : y Enter the data: 30

Do u want 2 add another node?(y/n) : n Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 2 Minimum Element is : 10

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 3 Maximum Element is : 30

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 4 10 20 30

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 5 10 20 30

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 6 Enter key to search: 20 Element found...

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 7 Enter key to delete: 10 Deleted Successfully...

Do u want to continue...?(y/n) : y

1. CREATE TREE
2. Find Minimum 3.Find Maximum 4.Print INORDER
3. Print PREORDER
4. Find Key 7.Deletenode 8.Exit

Enter the choice: 8

Do u want to continue...?(y/n) : n

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 5

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

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

============================================================================

*Name: Assignment\_5.cpp Author: Sahil Deshmukhi*

============================================================================\*/

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

struct node

{

char data; node \*left; node \*right;

};

class tree

{

char prefix[30];

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; /\*stack s1 is being used for flag . A NULL data implies that the right subtree has not been visited \*/

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;

/\* 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:-

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 : +

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 6

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**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.

============================================================================

*Name : Assignment\_6.cpp Author : Sahil Deshmukh*

============================================================================\*/

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

#define TRUE 1

#define FALSE 0

// declaring an adjacency list for storing the graph class lgra

{

private:

struct node1

{

int vertex;

struct node1 \*next;

};

node1 \*head[MAX]; int visited[MAX];

public:

//static int nodecount; lgra();

void create(); void dfs(int);

};

//constructor lgra::lgra()

{

int v1;

for (v1 = 0; v1 < MAX; v1++)

visited[v1] = FALSE;

for (v1 = 0; v1 < MAX; v1++)

head[v1] = NULL;

}

void lgra::create()

{

int v1, v2; char ans;

node1 \*N, \*first;

cout << "Enter the vertices no. beginning with 0 : "; do

{

cout << "\nEnter the Edge of a graph : \n"; cin >> v1 >> v2;

if (v1 >= MAX || v2 >= MAX)

cout << "Invalid Vertex Value!!\n"; else

{

//creating link from v1 to v2 N = new node1;

if (N == NULL)

cout << "Insufficient Memory!!\n"; N->vertex = v2;

N->next = NULL; first = head[v1]; if (first == NULL)

head[v1] = N; else

{

while (first->next != NULL) first = first->next;

first->next = N;

}

//creating link from v2 to v1 N = new node1;

if (N == NULL)

cout << "Insufficient Memory!!\n"; N->vertex = v1;

N->next = NULL; first = head[v2]; if (first == NULL)

head[v2] = N; else

{

while (first->next != NULL) first = first->next;

first->next = N;

}

}

cout << "\n Want to add more edges?(y/n) : "; cin >> ans;

} while (ans == 'y');

}

//dfs function

void lgra::dfs(int v1)

{

node1 \*first; cout << endl

<< v1;

visited[v1] = TRUE; first = head[v1]; while (first != NULL)

if (visited[first->vertex] == FALSE) dfs(first->vertex);

else

first = first->next;

}

int main()

{

int v1; lgra g;

g.create();

cout << endl << "Enter the vertex from where you want to traverse : "; cin >> v1;

if (v1 >= MAX)

cout << "Invalid Vertex!!\n"; else

{

cout << "The Dfs of the graph : "; g.dfs(v1);

}

return 0;

}

**Output**

Enter the vertices no. beginning with 0 : Enter the Edge of a graph :

4

2

Want to add more edges?(y/n) : y Enter the Edge of a graph :

1

3

Want to add more edges?(y/n) : n

Enter the vertex from where you want to traverse : 4 The Dfs of the graph :

4

2

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

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 7

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## PROBLEM STATEMENT:

**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 minimum total cost.Solve the problem by suggesting appropriate data structures.**

============================================================================

*Name : Assignment\_7.cpp Author : Sahil Deshmukh*

====================================================================

\*/

#include<iostream> using namespace std;

class tree

{

public:

};

int a[20][20],l,u,w,i,j,v,e,visited[20];

void input(); void display(); void minimum();

void tree::input()

{

cout<<"Enter the no. of branches: "; cin>>v;

for(i=0;i<v;i++)

{

visited[i]=0; for(j=0;j<v;j++)

{

a[i][j]=999;

}

}

cout<<"\nEnter the no. of connections: "; cin>>e;

for(i=0;i<e;i++)

{

cout<<"Enter the end branches of connections: "<<endl; cin>>l>>u;

cout<<"Enter the phone company charges for this connection: "; cin>>w;

a[l-1][u-1]=a[u-1][l-1]=w;

}

}

void tree::display()

{

cout<<"\nAdjacency matrix:"; for(i=0;i<v;i++)

{

cout<<endl; for(j=0;j<v;j++)

{

cout<<a[i][j]<<" ";

}

cout<<endl;

}

}

void tree::minimum()

{

int p=0,q=0,total=0,min; visited[0]=1;

for(int count=0;count<(v-1);count++)

{

min=999; for(i=0;i<v;i++)

{

if(visited[i]==1)

{

for(j=0;j<v;j++)

{

if(visited[j]!=1)

{

if(min > a[i][j])

{

min=a[i][j]; p=i;

q=j;

}

}

}

}

}

visited[p]=1;

visited[q]=1; total=total+min;

cout<<"Minimum cost connection is"<<(p+1)<<" -> "<<(q+1)<<" with charge : "<<min<< endl;

}

cout<<"The minimum total cost of connections of all branches is: "<<total<<endl;

}

int main()

{

int ch; tree t; do

{

cout<<"==========PRIM'S ALGORITHM================="<<endl;

cout<<"\n1.INPUT\n \n2.DISPLAY\n \n3.MINIMUM\n"<<endl; cout<<"Enter your choice :"<<endl;

cin>>ch;

switch(ch)

{

case 1: cout<<"\*\*\*\*\*\*\*INPUT YOUR VALUES\*\*\*\*\*\*\*"<<endl; t.input();

break;

case 2: cout<<"\*\*\*\*\*\*\*DISPLAY THE CONTENTS\*\*\*\*\*\*\*\*"<<endl; t.display();

break;

case 3: cout<<"\*\*\*\*\*\*\*\*\*MINIMUM\*\*\*\*\*\*\*\*\*\*\*\*"<<endl; t.minimum();

break;

}

}while(ch!=4); return 0;

}

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 8

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**Consider a scenario for Hospital to cater services to different kinds of patients as Serious (top priority), b) non-serious (medium priority), c) General Checkup (Least priority). Implement the priority queue to cater services to the patients.**

============================================================================

============================================================================\*/

#include<iostream> #include<string>

//#define N 20

using namespace std; string Q[10];

int Pr[10];

int r = -1,f = -1,n;

void enqueue(string data,int p)//Enqueue function to insert data and its priority in queue

{

int i;

if((f==0)&&(r==n-1)) //Check if Queue is full cout<<"Queue is full";

else {

if(f==-1) { //if Queue is empty 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 print() { //print the data of Queue int i;

for(i=f;i<=r;i++) {

cout << "Patient's Name - "<<Q[i]; switch(Pr[i]) {

case 1:

cout << " Priority - 'Checkup' " << endl;

break; case 2:

cout << " Priority - 'Non-serious' " << endl;

break; case 3:

cout << " Priority - 'Serious' " << endl;

break; default:

cout << "Priority not found" << endl;

}

}

}

void dequeue() { //remove the data from front if(f == -1) {

cout<<"Queue is Empty";

}

else {

cout<<"deleted Element ="<<Q[f]<<endl; cout<<"Its Priority = "<<Pr[f]<<endl;

if(f==r) f = r = -1; else f++;

}

}

int main() {

string data; int opt,i,p;

cout<<"Enter Your Choice:-"<<endl; do {

cout << "1 for Insert the Data in Queue" << endl << "2 for show the Data in Queue " << endl

<< "3 for Delete the data from the Queue"

<< endl << "0 for Exit"<< endl; cin >> opt;

switch(opt) {

case 1:

cout << "Enter the number of patinent" << endl; cin >> n;

i = 0;

for(i=0;i<n;i++)

{

cout << "Enter your name of the patient : "; cin >> data;

genral checkup) : ";

cout << "Enter your Prioritys (3: serious, 2: non-serious, 1: cin >> p;

enqueue(data,p);

}

break;

case 2:

print();

break; case 3:

break;

}

dequeue();

}while(opt!=0); return 0;

}

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 9

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**A Dictionary stores keywords and 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

============================================================================

============================================================================\*/

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

class dictionary; class node

{

string word, meaning; node \*left, \*right; public:

friend class dictionary; node ()

{

left = NULL; right = NULL;

}

node (string word, string meaning)

{

this->word = word;

this->meaning = meaning; left = NULL;

right = NULL;

}

};

class dictionary

{

node \*root; public:

dictionary ()

{

root = NULL;

}

void create ()

{

int n;

string wordI, meaningI;

cout << "\nHow many Word to insert?:\n"; cin >> n;

for (int i = 0; i < n; i++)

{

cout << "\nENter Word: "; cin >> wordI;

cout << "\nEnter Meaning: "; cin >> meaningI;

insert (wordI, meaningI);

}

}

void inorder\_rec (node \* rnode)

{

if (rnode)

{

inorder\_rec (rnode->left);

cout << " " << rnode->word << " : " << rnode->meaning << endl; inorder\_rec (rnode->right);

}

}

void postorder\_rec (node \* rnode)

{

if (rnode)

{

postorder\_rec (rnode->right);

cout << " " << rnode->word << " : " << rnode->meaning << endl; postorder\_rec (rnode->left);

}

}

void inorder ()

{

inorder\_rec (root);

}

void postorder ()

{

postorder\_rec (root);

}

bool insert (string word, string meaning)

{

node \*p = new node (word, meaning); if (root == NULL)

{

root = p; return true;

}

node \*cur = root; node \*par = root;

while (cur != NULL) //traversal

{

if (word > cur->word)

{

par = cur;

cur = cur->right;

}

else if (word < cur->word)

{

}

else

{

}

}

par = cur;

cur = cur->left;

cout << "\nWord is already in the dictionary."; return false;

if (word > par->word) //insertion of node

{

par->right = p; return true;

}

else

{

par->left = p; return true;

}

}

int search (string key)

{

node \*tmp = root; int count;

if (tmp == NULL)

{

return -1;

}

if (root->word == key) return 1;

while (tmp != NULL)

{

if ((tmp->word) > key)

{

tmp = tmp->left; count++;

}

else if ((tmp->word) < key)

{

tmp = tmp->right; count++;

}

else if (tmp->word == key)

{

return ++count;

}

}

return -1;

}

};

int main ()

{

string word; dictionary months; months.create ();

cout << "Ascending order\n"; months.inorder ();

cout << "\nDescending order:\n"; months.postorder ();

cout << "\nEnter word to search: "; cin >> word;

int comparisons = months.search (word); if (comparisons == -1)

{

cout << "\nNot found word";

}

else

{

cout << "\n " << word << " found in " << comparisons << " comparisons";

}

return 0;

}

**Output**

How many Word to insert?:

4

ENter Word: pritam Enter Meaning: psg ENter Word: akshay Enter Meaning: aks ENter Word: pranjali Enter Meaning: pran ENter Word: sarthak

Enter Meaning: sart Ascending order akshay : aks pranjali : pran pritam : psg sarthak : sart

Descending order:

sarthak : sart pritam : psg pranjali : pran akshay : aks

Enter word to search: pranjali pranjali found in 3 comparisons

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

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 10

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**Implement the Heap sort algorithm implemented in Java demonstrating heap/shell data structure with modularity of programming language**

============================================================================

============================================================================\*/

package sidd\_a10; import java.util.\*;

public class Assignment\_10

{

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 && arr[left] > arr[i]) max = left;

if (right <= N && 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();

}

}

/\*/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

# Assignment 11

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////\*/

/\*

## Problem Statement:

**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,ch1,rec,start,count,add,n1,add2,start2,n2,y,a,b,on,oname,add3,start3,n3,y1,add4,start4,n4; char name[20],name2[20];

tel t1; count=0; fstream g,f; do

{

cout<<"\n>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<";

cout<<"\n1.Insert and overwrite\n2.Show\n3.Search & Edit(number)\n4.Search & Edit(name)\n5.Search & Edit(onlynumber)\n6.Search & 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:t1.accept();

f.write((char\*) &t1,(sizeof(t1)));

cout<<"\nDo you want to enter more records?\n1.Yes\n2.No"; cin>>ch1;

if(ch1==1) goto x;

else

{

}

case 2:

f.close(); break;

f.open("StuRecord.txt",ios::in);

f.read((char\*) &t1,(sizeof(t1)));

//cout<<"\n\tRoll No.\t\tName \t\t Division \t\t Address"; while(f)

{

t1.show();

f.read((char\*) &t1,(sizeof(t1)));

}

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\*)&t1,(sizeof(t1)));

while(f)

{

if(rec==t1.rollNo)

{

cout<<"\nRecord found"; add=f.tellg(); f.seekg(0,ios::beg); start=f.tellg();

n1=(add-start)/(sizeof(t1)); f.seekp((n1-1)\*sizeof(t1),ios::beg); t1.accept();

f.write((char\*) &t1,(sizeof(t1))); f.close();

count++; break;

}

f.read((char\*)&t1,(sizeof(t1)));

}

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\*)&t1,(sizeof(t1)));

while(f)

{

y=(strcmp(name,t1.name)); if(y==0)

{

cout<<"\nName found"; add2=f.tellg(); f.seekg(0,ios::beg); start2=f.tellg();

n2=(add2-start2)/(sizeof(t1)); f.seekp((n2-1)\*sizeof(t1),ios::beg); t1.accept();

f.write((char\*) &t1,(sizeof(t1))); f.close();

break;

}

f.read((char\*)&t1,(sizeof(t1)));

}

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\*) &t1,(sizeof(t1)));

while(f)

{

if(on==t1.rollNo)

{

cout<<"\n\tNumber found"; add3=f.tellg(); f.seekg(0,ios::beg); start3=f.tellg();

n3=(add3-start3)/(sizeof(t1)); f.seekp((n3-1)\*(sizeof(t1)),ios::beg); t1.accept2(); f.write((char\*)&t1,(sizeof(t1))); f.close();

break;

}

f.read((char\*)&t1,(sizeof(t1)));

}

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\*)&t1,(sizeof(t1)));

while(f)

{

y1=(strcmp(name2,t1.name)); if(y1==0)

{

cout<<"\nName found"; add4=f.tellg(); f.seekg(0,ios::beg); start4=f.tellg();

n4=(add4-start4)/(sizeof(t1)); f.seekp((n4-1)\*sizeof(t1),ios::beg); t1.accept3();

f.write((char\*) &t1,(sizeof(t1))); f.close();

break;

}

f.read((char\*)&t1,(sizeof(t1)));

}

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 \*)&t1,sizeof(t1)); while(!f.eof())

{

if (t1.getRollNo() != roll) g.write((char \*)&t1,sizeof(t1)); f.read((char \*)&t1,sizeof(t1));

}

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** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

* 1. Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice :1 Enter Roll Number : 1 Enter the Name : Dipak Enter the Division:A

Enter the Address:Satpur

Do you want to enter morerecords? 1.Yes

2.No1

Enter Roll Number : 28 Enter the Name : Rutesh Enter the Division:A

Enter the Address:Amrutdham

Do you want to enter morerecords? 1.Yes

2.No1

Enter Roll Number : 33 Enter the Name : shekhar Enter the Division:A Enter the Address:sidko

Do you want to enter morerecords? 1.Yes

2.No2

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

1.Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice :2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Dipak | A | tSatpur |
| 28 | Rutesh | A | tAmrutdham |
| 33 | shekhar | A | tsidko |

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

1.Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice :3

Enter the roll number you want to find 1 Record found

Enter Roll Number : 1 Enter the Name : dipak Enter the Division:A Enter the Address:satpur

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

1.Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice : 4

Enter the name you want to find and edit Rutesh Name found

Enter Roll Number : 25 Enter the Name : RUTESH Enter the Division: A

Enter the Address: Adgaoun

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

1.Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice :7

Please Enter the Roll No. of Student WhoseInfo You Want to Delete: 33 The record with the roll no. 33 hasbeen deleted

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<

1.Insert and overwrite 2.Show

3.Search &Edit(number) 4.Search & Edit(name) 5.Search &Edit(onlynumber) 6.Search & edit(only name) 7.Delete a Student Record 8.Exit

Enter theChoice :8

Thank you

Process exited after 240.9 seconds with return value 0 Press any key to continue . . .