Experiment 6 : Three list are maintained with the integer roll nos of the students. One list contains the roll nos and name of the students, second list contains the roll nos and the course name (BE, BCA etc) for the student and the third list contains the roll no. and CGPA of the student. Find a way to check the common roll nos in the 3 lists. Note: the 3 lists are sorted by the roll nos.

Cos attained:

CO2, CO3 and CO4

#include<bits/stdc++.h>

using namespace std;

int isPresent(vector<int>&arr, int k){

int n = arr.size();

int l=0, r=n-1, ans=0;

while(l<=r){

int m = (l+r)/2;

if(arr[m]==k){

ans = 1;

break;

}else if(arr[m]<k){

l=m+1;

}else{

r=m-1;

}

}

return ans;

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<"";

}

cout<<"\n";

}

int main(){

vector<int>v1 = {1,2,3,4,5,8,9};

vector<int>v2 = {2,4,8,9,11,13};

vector<int>v3 = {1,2,3,5,9,12,18,18};

vector<int>ans;

int n1=v1.size(), n2=v2.size(), n3=v3.size();

int n = min({n1,n2,n3});

if(n1==n){

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

if(isPresent(v2,v1[i]) && isPresent(v3,v1[i])){

ans.push\_back(v1[i]);

}

}

}else if(n2==n){

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

if(isPresent(v1,v2[i]) && isPresent(v3,v2[i])){

ans.push\_back(v2[i]);

}

}

}else{

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

if(isPresent(v2,v3[i]) && isPresent(v1,v3[i])){

ans.push\_back(v3[i]);

}

}

}

print1d(ans);

return 0;

}

Experiment 7 : A set of strings are saved in a data structure. Represent the strings in form a non linear data structure in such a way that the searching takes the minimal time

CO attained: CO2, CO4

#include <bits/stdc++.h>

using namespace std;

class Node{

public:

Node \*alpha[26] = {0};

bool end = false;

bool containsKey(char ch){

return alpha[ch-'a'] != NULL;

}

void setNode(char ch, Node \*node){

alpha[ch-'a'] = node;

}

Node \*getNode(char ch){

return alpha[ch-'a'];

}

void setEnd(){

end = true;

}

bool isEnd(){

return end;

}

};

class Trie{

Node \*root;

public:

Trie(){

root = new Node();

}

void insertWord(string s){

Node \*n1 = root;

int n = s.size();

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

if( !(n1->containsKey(s[i])) ){

n1->setNode(s[i],new Node());

}

n1 = n1->getNode(s[i]);

}

n1->setEnd();

}

bool searchWord(string s){

int n = s.size();

Node \*n1 = root;

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

if( !(n1->containsKey(s[i]))){

return false;

}

n1 = n1->getNode(s[i]);

}

return n1->isEnd();

}

};

int main()

{

Trie t1;

t1.insertWord("Hello");

t1.insertWord("Hell");

t1.insertWord("Heed");

t1.insertWord("Heap");

t1.insertWord("Help");

t1.insertWord("Helo");

cout<<t1.searchWord("Hello")<<"\n";

cout<<t1.searchWord("Hell");

return 0;

}