1. 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.
2. 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
3. Without comparing the integer values in an array, sort the array of integers in ascending order.
4. List of Marks obtained by students of your class is maintained in form a linked List. Apply a sorting algorithm with complexity O (n^2) to sort the list.
5. Two arrays are maintained such that the marks are in descending order. We need to reverse the two arrays using another data structure so that the arrays are in ascending order. Hint: Use LIFO

Solution:

1.

#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;

}

2.

#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;

}

3.

#include <bits/stdc++.h>

using namespace std;

void count\_sort(vector<int>&v){

map<int,int>um;

for(auto &x : v){

um[x]++;

}

int itr = 0;

for(auto &x : um){

int num = x.first;

int times = x.second;

while(times--){

v[itr++] = num;

}

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<" ";

}

cout<<"\n";

}

int main()

{

vector<int>v = {2,5,3,4,1,6,7,3,5,9};

print1d(v);

count\_sort(v);

print1d(v);

return 0;

}

4.

#include<bits/stdc++.h>

using namespace std;

class Node{

public:

int val;

Node \*next;

Node(){

this->val = INT\_MIN;

this->next = NULL;

}

Node(int x){

val = x;

this->next = NULL;

}

};

class LL{

Node \*head;

public:

LL(){

this->head = NULL;

}

void insertAtBegin(int x){

Node \*n1 = new Node(x);

n1->next = head;

head = n1;

}

void insertAtEnd(int x){

Node \*n1 = new Node(x);

if(head==NULL){

head=n1;

return ;

}

Node \*temp = head;

while(temp->next){

temp = temp->next;

}

temp->next = n1;

}

void sorting(){

Node \*temp1 = head, \*temp2 = head;

while(temp1->next){

temp2 = temp1->next;

while(temp2){

if(temp2->val < temp1->val){

int temp = temp2->val;

temp2->val = temp1->val;

temp1->val = temp;

}

temp2 = temp2->next;

}

temp1 = temp1->next;

}

}

void printList(){

if(!head){

cout<<"Underflow: No data to print.";

return ;

}

Node \*temp = head;

while(temp){

cout<<temp->val<<"->";

temp = temp->next;

}

}

};

int main(){

LL l1;

l1.insertAtEnd(11); // 11

l1.insertAtEnd(12); // 11 12

l1.insertAtEnd(13); // 11 12 13

l1.insertAtEnd(14); // 11 12 13 14

l1.insertAtBegin(90); // 90 11 12 13 14 15

l1.insertAtBegin(80); // 80 90 11 12 13 14 15

l1.sorting();

l1.printList();cout<<"\n"; // 11->13->14

return 0;

}

5,

#include<bits/stdc++.h>

using namespace std;

void reverse\_using\_stack(vector<int>&v){

stack<int>st;

for(auto &x : v){

st.push(x);

}

int itr=0;

while(st.size()){

v[itr++] = st.top();

st.pop();

}

}

void print1d(vector<int>&v){

for(auto &x : v){

cout<<x<<" ";

}

cout<<"\n";

}

int main(){

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

vector<int>v2 = {9,8,7,6,5};

reverse\_using\_stack(v1);

reverse\_using\_stack(v2);

print1d(v1);

print1d(v2);

return 0;

}