<https://programs.programmingoneonone.com/2021/08/leetcode-maximum-subarray-problem-solution.html>

int maxSubArray(int\* nums, int n) {

int ans=nums[0],i,sum=0;

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

sum+=nums[i];

if(ans<sum)ans=sum;

if(sum<0)sum=0;

}

return ans;

}

<https://www.geeksforgeeks.org/chocolate-distribution-problem/>

// C program to solve chocolate distribution problem

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

// Compare function for qsort

int cmpfunc(const void\* a, const void\* b)

{

return (\*(int\*)a - \*(int\*)b);

}

// arr[0..n-1] represents sizes of packets

// m is number of students.

// Returns minimum difference between maximum

// and minimum values of distribution.

int findMinDiff(int arr[], int n, int m)

{

// if there are no chocolates or number

// of students is 0

if (m == 0 || n == 0)

return 0;

// Sort the given packets

qsort(arr, n, sizeof(int), cmpfunc);

// Number of students cannot be more than

// number of packets

if (n < m)

return -1;

// Largest number of chocolates

int min\_diff = INT\_MAX;

// Find the subarray of size m such that

// difference between last (maximum in case

// of sorted) and first (minimum in case of

// sorted) elements of subarray is minimum.

for (int i = 0; i + m - 1 < n; i++) {

int diff = arr[i + m - 1] - arr[i];

if (diff < min\_diff)

min\_diff = diff;

}

return min\_diff;

}

int main()

{

int arr[] = { 12, 4, 7, 9, 2, 23, 25, 41, 30,

40, 28, 42, 30, 44, 48, 43, 50 };

int m = 7; // Number of students

int n = sizeof(arr) / sizeof(arr[0]);

printf("Minimum difference is %d",

findMinDiff(arr, n, m));

return 0;

}

// This code is contributed by Aditya Kumar (adityakumar129)

**30-07-2022**

**NEXT PERMUTATION**

**#include <iostream>**

**#include<vector>**

**using namespace::std;**

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**int main() {**

**vector<int>a={1,2,3};**

**int n=a.size();**

**int i,idx=-1;**

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

**if(a[i]>a[i-1]){**

**idx=i;**

**break;**

**}**

**}**

**if(idx==-1)**

**reverse(a.begin(),a.end());**

**else{**

**int p=idx;**

**for(i=idx+1;i<n;i++){**

**if(a[i]>a[idx-1]&&a[i]<=a[p])**

**p=i;**

**}**

**swap(a[idx-1],a[p]);**

**reverse(a.begin()+idx,a.end());**

**}**

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

**cout<<a[i]<<endl;**

**}**

<https://www.interviewbit.com/problems/repeat-and-missing-number-array/>

#include <stdio.h>

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int main(void) {

int a[100]={3,1,2,5,3};

int i,n=5,fq[100];

int x;

int m1,m2;

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

x=abs(a[i])-1;

if(a[x]<0)

m1=x+1;

else

a[x]\*=-1;

}

printf("%d ",m1);

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

if(a[i]>0)

m2=i+1;

}

printf("%d ",m2);

return 0;

}

**KTH LARGEST AND SMALLEST ELEMENT IN ARRAY**

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int sort(int a[],int n){

int i,j,t;

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

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

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

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

return 0;

}

int main()

{

int i,n,a[100],d,j,max=a[0],fq[100],c=1,k=3;

scanf("%d",&n);

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

scanf("%d",&a[i]);

sort(a,n);

printf("%d ",a[n-k]);

return 0;

**}**

**TRAPPING RAIN WATER**

<https://leetcode.com/problems/trapping-rain-water/>

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int sort(int a[],int n){

int i,j,t;

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

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

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

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

return 0;

}

int max(int a,int b){

if(a>b)

return a;

else

return b;

}

int min(int a,int b){

if(a>b)

return b;

else

return a;

}

int main()

{

int i,n,a[100],j,left,right,res=0;

scanf("%d",&n);

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

scanf("%d",&a[i]);

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

left=a[i];

for(j=0;j<i;j++){

left=max(left,a[j]);

}

right=a[i];

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

right=max(right,a[j]);

}

res+=(min(right,left)-a[i]);

}

printf("%d\n",res);

return 0;

}

<https://leetcode.com/problems/product-of-array-except-self/>

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int sort(int a[],int n){

int i,j,t;

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

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

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

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

return 0;

}

int max(int a,int b){

if(a>b)

return a;

else

return b;

}

int min(int a,int b){

if(a>b)

return b;

else

return a;

}

int main()

{

int i,j,n,a[100],b[100],p;

scanf("%d",&n);

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

scanf("%d",&a[i]);

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

p=1;

for(j=0;j<n;j++){

p\*=(a[j]);

}

p/=a[i];

b[i]=p;

}

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

printf("%d\n",b[i]);

return 0;

}

<https://leetcode.com/problems/3sum/>

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int sort(int a[],int n){

int i,j,t;

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

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

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

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

return 0;

}

int max(int a,int b){

if(a>b)

return a;

else

return b;

}

int min(int a,int b){

if(a>b)

return b;

else

return a;

}

int main()

{

int i,j,n=6,k,a[100],d,sum;

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

scanf("%d",&a[i]);

for(i=0;i<n-2;i++){

for(j=i+1;j<n-1;j++){

sum=a[i]+a[j];

for(k=j+1;k<n;k++){

if((sum+a[k])==0)

printf("%d %d %d\n",a[i],a[j],a[k]);

}

}

}

return 0;

}

**CONTAINER WITH MOST WATER**

<https://leetcode.com/problems/container-with-most-water/>

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

int sort(int a[],int n){

int i,j,t;

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

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

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

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

}

return 0;

}

int max(int a,int b){

if(a>b)

return a;

else

return b;

}

int min(int a,int b){

if(a>b)

return b;

else

return a;

}

int main()

{

int i,j,n=9,k,a[100],d,sum;

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

scanf("%d",&a[i]);

int l=0;

int r=n-1;

int w=0;

while(l<r){

w=max(w,min(a[l],a[r])\*(r-l));

if(a[l]>a[r])r--;

else

l++;

}

printf("%d\n",w);

return 0;

}

**BIGGEST NUMBER**

<https://www.geeksforgeeks.org/given-an-array-of-numbers-arrange-the-numbers-to-form-the-biggest-number/>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

bool cmp(string &x,string &y){

string xy=x+y;

string yx=y+x;

return xy>yx;

}

string larg(vector<int> a){

int n=a.size();

vector<string> b;

int i;

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

b.push\_back(to\_string(a[i]));

}

sort(b.begin(),b.end(),cmp);

string s="";

for(auto val:b){

s+=val;

}

return s;

}

int main() {

vector<int>a={3, 30, 34, 5,9};

string l=larg(a);

cout<<l<<endl;

}

**LINKED LIST;**

**Detecting a cycle**

#include <stdio.h>

#include<math.h>

#include<string.h>

struct node{

int data;

struct node\*link;

};

void dis(struct node\*head){

struct node\*p=head;

while(p!=NULL){

printf("%d ",p->data);

p=p->link;

}

}

int detectloop(struct node \*\*list)

{

struct node \*S = \*list, \*F=\*list;

while( F!=NULL && F->link!=NULL)

{

S=S->link;

F=F->link->link;

if(F==S)

{

printf("loop exists");

return 1;

}

}

return 0;

}

int main(){

struct node\*head=(struct node\*)malloc(sizeof(struct node\*));

head=NULL;

struct node\*f1=(struct node\*)malloc(sizeof(struct node\*));

struct node\*f2=(struct node\*)malloc(sizeof(struct node\*));

struct node\*f3=(struct node\*)malloc(sizeof(struct node\*));

struct node\*f4=(struct node\*)malloc(sizeof(struct node\*));

struct node\*f5=(struct node\*)malloc(sizeof(struct node\*));

head=f1;

f1->data=10;

f1->link=f2;

f2->data=20;

f2->link=f3;

f3->data=30;

f3->link=f4;

f4->data=40;

f4->link=f5;

f5->data=50; f5->link=f2;

printf("%d\n",detectloop(&f1));

return 0;

}

**MERGING TWO SORTED LINKED LIST**

**#include <stdio.h>**

**#include<math.h>**

**#include<string.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**struct node\*r;**

**void ins(struct node\*\*head){**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**int v;**

**printf("enter v\n");**

**scanf("%d",&v);**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**void dis2(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**struct node\*mer(struct node \*\*head1,struct node \*\*head2){**

**struct node\*p1=\*head1;**

**struct node\*p2=\*head2;**

**while(p1->link!=NULL)**

**p1=p1->link;**

**p1->link=p2;**

**while(p2->link!=NULL)**

**p2=p2->link;**

**}**

**int main(){**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1)**

**ins(&head1);**

**else if(c==2)**

**ins(&head2);**

**else if(c==3)**

**dis(&head1);**

**else if(c==4)**

**dis(&head2);**

**else if(c==5){**

**mer(&head1,&head2);**

**dis(&r);**

**}**

**}**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/multiply-two-numbers-represented-linked-lists/**](https://www.geeksforgeeks.org/multiply-two-numbers-represented-linked-lists/)

**#include <stdio.h>**

**#include<math.h>**

**#include<string.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**struct node\*r;**

**void ins(struct node\*\*head){**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**int v;**

**printf("enter v\n");**

**scanf("%d",&v);**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**void dis2(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**int \*pro(struct node\*\*head1,struct node\*\*head2){**

**int num1=0,num2=0;**

**struct node\*p1=\*head1;**

**struct node\*p2=\*head2;**

**while(p1!=NULL||p2!=NULL){**

**while(p1!=NULL){**

**num1=(num1\*10)+(p1->data);**

**p1=p1->link;**

**}**

**while(p2!=NULL){**

**num2=(num2\*10)+(p2->data);**

**p2=p2->link;**

**}**

**}**

**return num1\*num2;**

**}**

**int main(){**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1)**

**ins(&head1);**

**else if(c==2)**

**ins(&head2);**

**else if(c==3)**

**dis(&head1);**

**else if(c==4)**

**dis(&head2);**

**else if(c==5){**

**printf("%d\n",pro(&head1,&head2));**

**}**

**}**

**return 0;**

**}**

[**https://leetcode.com/problems/reorder-list/**](https://leetcode.com/problems/reorder-list/)

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**void ins(struct node\*\*head){**

**int v;**

**printf("enter v\n");**

**scanf("%d",&v);**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**struct node\*pblm(struct node\*\*head){**

**struct node\*p1=\*head;**

**struct node\*p2=\*head;**

**while(p1!=NULL&&p1->link!=NULL){**

**p1=p1->link->link;**

**p2=p2->link;**

**}**

**struct node\*f=\*head;**

**struct node\*s=p2->link;**

**p2->link=NULL;**

**while(s!=NULL){**

**struct node\*t1=f->link;**

**struct node\*t2=s->link;**

**f->link=s;**

**s->link=t1;**

**f=t1;**

**s=t2;**

**}**

**dis(&f);**

**dis(&s);**

**}**

**int main()**

**{**

**struct node\*head=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1)**

**ins(&head);**

**else if(c==2)**

**dis(&head);**

**else if(c==3){**

**pblm(&head);**

**}**

**}**

**return 0;**

**}**

**DELETING DUPLICATE ELEMENTS**

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**void ins(struct node\*\*head){**

**int v;**

**printf("enter v\n");**

**scanf("%d",&v);**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**void del(struct node\*\*head,int v){**

**struct node\*p1=\*head;**

**struct node\*p2;**

**while(p1!=NULL&&p1->data!=v){**

**p2=p1;**

**p1=p1->link;**

**}**

**p2->link=p1->link;**

**free(p1);**

**}**

**struct node\*del\_dup(struct node\*\*head){**

**struct node\*p1=\*head;**

**struct node\*p2;**

**struct node\*temp;**

**while(p1!=NULL&&p1->link!=NULL){**

**p2=p1;**

**while(p2->link!=NULL){**

**if(p1->data==p2->link->data){**

**temp=p2->link;**

**p2->link=p2->link->link;**

**free(p2);**

**}else**

**p2=p2->link;**

**}**

**p1=p1->link;**

**}**

**}**

**int order(struct node\*\*head){**

**int t,f=1;**

**struct node\*p=\*head;**

**while(p->link!=NULL){**

**if(f==1){**

**if(p->data>p->link->data){**

**t=p->data;**

**p->data=p->link->data;**

**p->link->data=t;**

**}**

**}**

**else{**

**if(p->data<p->link->data){**

**t=p->data;**

**p->data=p->link->data;**

**p->link->data=t;**

**}**

**}**

**p=p->link;**

**f=!f;**

**}**

**}**

**int main()**

**{**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1)**

**ins(&head1);**

**else if(c==2)**

**dis(&head1);**

**else if(c==3){**

**del\_dup(&head1);**

**}**

**else if(c==4)**

**order(&head1);**

**}**

**return 0;**

**}**

**REMOVING LOOP**

**#include <stdio.h>**

**#include<math.h>**

**#include<string.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**int removeloop(struct node \*\*list)**

**{**

**struct node \*S = \*list, \*F=\*list;**

**while( F!=NULL && F->link!=NULL)**

**{**

**S=S->link;**

**F=F->link->link;**

**if(F==S)**

**{**

**break;**

**}**

**}**

**if(S==F){**

**S=\*list;**

**if(S==F){**

**while(F->link!=S)F=F->link;**

**}else{**

**while(S->link!=F->link){**

**S=S->link;**

**F=F->link;**

**}**

**}**

**F->link=NULL;**

**}**

**return 0;**

**}**

**int main(){**

**struct node\*head=(struct node\*)malloc(sizeof(struct node\*));**

**head=NULL;**

**struct node\*f1=(struct node\*)malloc(sizeof(struct node\*));**

**struct node\*f2=(struct node\*)malloc(sizeof(struct node\*));**

**struct node\*f3=(struct node\*)malloc(sizeof(struct node\*));**

**struct node\*f4=(struct node\*)malloc(sizeof(struct node\*));**

**struct node\*f5=(struct node\*)malloc(sizeof(struct node\*));**

**head=f1;**

**f1->data=10;**

**f1->link=f2;**

**f2->data=20;**

**f2->link=f3;**

**f3->data=30;**

**f3->link=f4;**

**f4->data=40;**

**f4->link=f5;**

**f5->data=50; f5->link=f2;**

**removeloop(&head);**

**dis(&head);**

**return 0;**

**}**

**31-07-2022**

**Intersection point**

[**https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1**](https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1)

**// C program to get intersection point of two linked list**

**#include <stdio.h>**

**#include <stdlib.h>**

**/\* Link list node \*/**

**typedef struct Node {**

**int data;**

**struct Node\* next;**

**} Node;**

**/\* function to get the intersection point of two linked**

**lists head1 and head2 \*/**

**int len(Node\*head){**

**int l;**

**Node\*temp=head;**

**while(temp!=NULL){**

**temp=temp->next;**

**l++;**

**}**

**return l;**

**}**

**int getIntesectionNode(Node\* head1, Node\* head2)**

**{**

**int l1=len(head1);**

**int l2=len(head2);**

**int d=0;**

**Node\*p1;**

**Node\*p2;**

**if(l1>l2){**

**d=l1-l2;**

**p1=head1;**

**p2=head2;**

**}else{**

**d=l2-l1;**

**p1=head2;**

**p2=head1;**

**}**

**while(d){**

**p1=p1->next;**

**if(p1==NULL)**

**return -1;**

**d--;**

**}**

**while(p1!=NULL&&p2!=NULL){**

**if(p1==p2)**

**return p1->data;**

**p1=p1->next;**

**p2=p2->next;**

**}**

**}**

**// Driver Code**

**int main()**

**{**

**/\***

**Create two linked lists**

**1st 3->6->9->15->30**

**2nd 10->15->30**

**15 is the intersection point**

**\*/**

**Node\* newNode;**

**// Addition of new nodes**

**Node\* head1 = (Node\*)malloc(sizeof(Node));**

**head1->data = 10;**

**Node\* head2 = (Node\*)malloc(sizeof(Node));**

**head2->data = 3;**

**newNode = (Node\*)malloc(sizeof(Node));**

**newNode->data = 6;**

**head2->next = newNode;**

**newNode = (Node\*)malloc(sizeof(Node));**

**newNode->data = 9;**

**head2->next->next = newNode;**

**newNode = (Node\*)malloc(sizeof(Node));**

**newNode->data = 15;**

**head1->next = newNode;**

**head2->next->next->next = newNode;**

**newNode = (Node\*)malloc(sizeof(Node));**

**newNode->data = 30;**

**head1->next->next = newNode;**

**head1->next->next->next = NULL;**

**int intersectionPoint**

**= getIntesectionNode(head1, head2);**

**if (intersectionPoint==-1)**

**printf(" No Intersection Point \n");**

**else**

**printf("Intersection Point: %d\n",**

**intersectionPoint);**

**}**

[**https://www.geeksforgeeks.org/linked-list-in-zig-zag-fashion/**](https://www.geeksforgeeks.org/linked-list-in-zig-zag-fashion/)

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**void ins(struct node\*\*head){**

**int v;**

**printf("enter v\n");**

**scanf("%d",&v);**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**void del(struct node\*\*head,int v){**

**struct node\*p1=\*head;**

**struct node\*p2;**

**while(p1!=NULL&&p1->data!=v){**

**p2=p1;**

**p1=p1->link;**

**}**

**p2->link=p1->link;**

**free(p1);**

**}**

**struct node\*del\_dup(struct node\*\*head){**

**struct node\*p1=\*head;**

**struct node\*p2;**

**struct node\*temp;**

**while(p1!=NULL&&p1->link!=NULL){**

**p2=p1;**

**while(p2->link!=NULL){**

**if(p1->data==p2->link->data){**

**temp=p2->link;**

**p2->link=p2->link->link;**

**free(p2);**

**}else**

**p2=p2->link;**

**}**

**p1=p1->link;**

**}**

**}**

**int order(struct node\*\*head){**

**int t,f=1;**

**struct node\*p=\*head;**

**while(p->link!=NULL){**

**if(f==1){**

**if(p->data>p->link->data){**

**t=p->data;**

**p->data=p->link->data;**

**p->link->data=t;**

**}**

**}**

**else{**

**if(p->data<p->link->data){**

**t=p->data;**

**p->data=p->link->data;**

**p->link->data=t;**

**}**

**}**

**p=p->link;**

**f=!f;**

**}**

**}**

**int main()**

**{**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1)**

**ins(&head1);**

**else if(c==2)**

**dis(&head1);**

**else if(c==3){**

**del\_dup(&head1);**

**}**

**else if(c==4)**

**order(&head1);**

**}**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/rearrange-a-given-linked-list-in-place/**](https://www.geeksforgeeks.org/rearrange-a-given-linked-list-in-place/)

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**void ins(struct node\*\*head,int v){**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**struct node\*rev(struct node\*\*head){**

**struct node\*p=NULL;**

**struct node\*c=\*head;**

**struct node\*n=NULL;**

**while(c!=NULL){**

**n=c->link;**

**c->link=p;**

**p=c;**

**c=n;**

**}**

**\*head=p;**

**return \*head;**

**}**

**struct node\*ans(struct node\*\*head){**

**struct node\*p=\*head;**

**struct node\*p1;**

**struct node\*p2=\*head;**

**p1=p2->link;**

**while(p1!=NULL&&p1->link!=NULL){**

**p1=p1->link->link;**

**p2=p2->link;**

**}**

**struct node\*head1=\*head;**

**struct node\*head2=p2->link;**

**p2->link=NULL;**

**rev(&head2);**

**\*head=NULL;**

**struct node\*c=\*head;**

**while(head1!=NULL||head2!=NULL){**

**ins(&c,head1->data);**

**ins(&c,head2->data);**

**head1=head1->link;**

**head2=head2->link;**

**}**

**dis(&c);**

**}**

**int main()**

**{**

**struct node\*head=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1){**

**ins(&head,10);**

**ins(&head,20);**

**ins(&head,30);**

**ins(&head,40);**

**}**

**else if(c==2)**

**dis(&head);**

**else if(c==3){**

**ans(&head);**

**}**

**}**

**return 0;**

**}**

[**https://practice.geeksforgeeks.org/problems/segregate-even-and-odd-nodes-in-a-linked-list5035/1**](https://practice.geeksforgeeks.org/problems/segregate-even-and-odd-nodes-in-a-linked-list5035/1)

**#include <stdio.h>**

**#include <stdlib.h>**

**struct node{**

**int data;**

**struct node\*link;**

**};**

**struct node\*head1=NULL;**

**struct node\*head2=NULL;**

**void ins(struct node\*\*head,int v){**

**struct node\*p=\*head;**

**struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));**

**temp->data=v;**

**temp->link=NULL;**

**if(p==NULL)**

**\*head=temp;**

**else{**

**while(p->link!=NULL)**

**p=p->link;**

**p->link=temp;**

**}**

**}**

**void dis(struct node\*\*head){**

**struct node\*p=\*head;**

**while(p!=NULL){**

**printf("%d ",p->data);**

**p=p->link;**

**}**

**}**

**struct node\*rev(struct node\*\*head){**

**struct node\*p=NULL;**

**struct node\*c=\*head;**

**struct node\*n=NULL;**

**while(c!=NULL){**

**n=c->link;**

**c->link=p;**

**p=c;**

**c=n;**

**}**

**\*head=p;**

**return \*head;**

**}**

**int seg(struct node\*\*head){**

**struct node\*p1=\*head;**

**struct node\*p2=NULL;**

**struct node\*p3=NULL;**

**while(p1!=NULL){**

**if(p1->data%2==0)**

**ins(&p2,p1->data);**

**else**

**ins(&p3,p1->data);**

**p1=p1->link;**

**}**

**dis(&p2);**

**dis(&p3);**

**return 0;**

**}**

**int main()**

**{**

**struct node\*head=NULL;**

**int c;**

**while(1){**

**printf("enter c\n");**

**scanf("%d",&c);**

**if(c==1){**

**ins(&head,1);**

**ins(&head,2);**

**ins(&head,3);**

**ins(&head,4);**

**}**

**else if(c==2)**

**dis(&head);**

**else if(c==3){**

**seg(&head);**

**}**

**}**

**return 0;**

**}**

**1-08-2022**

[**https://www.geeksforgeeks.org/implement-two-stacks-in-an-array/**](https://www.geeksforgeeks.org/implement-two-stacks-in-an-array/)

**#include <stdio.h>**

**#include<math.h>**

**struct stack{**

**int t;**

**int s;**

**int \*arr;**

**};**

**int push(struct stack\*p,int v){**

**p->t++;**

**p->arr[p->t]=v;**

**}**

**int pop(struct stack\*p){**

**int v=p->arr[p->t];**

**p->t--;**

**}**

**int dis(struct stack\*p,int n){**

**int i;**

**for(i=n-1;i>=0;i--)**

**printf("%d ",p->arr[i]);**

**}**

**int main(void) {**

**struct stack\*p1=(struct stack\*)malloc(sizeof(struct stack\*));**

**p1->t=-1;**

**p1->s=5;**

**p1->arr=(int \*)malloc(p1->s\*sizeof(int ));**

**struct stack\*p2=(struct stack\*)malloc(sizeof(struct stack\*));**

**p2->t=-1;**

**p2->s=5;**

**p2->arr=(int \*)malloc(p2->s\*sizeof(int ));**

**int i,n,a[100];**

**scanf("%d",&n);**

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

**scanf("%d",&a[i]);**

**for(i=0;i<=n/2;i++)**

**push(p1,a[i]);**

**for(i=(n/2)+1;i<n;i++)**

**push(p2,a[i]);**

**dis(p1,n/2+1);**

**printf("\n");**

**dis(p2,n/2);**

**return 0;**

**}**

**Evaluating postfix**

[**https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/**](https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/)

**#include <stdio.h>**

**#include<math.h>**

**#include<stdlib.h>**

**#include<ctype.h>**

**#include<string.h>**

**struct stack{**

**int t;**

**int s;**

**char \*arr;**

**};**

**int push(struct stack\*p,char v){**

**p->t++;**

**p->arr[p->t]=v;**

**}**

**int pop(struct stack\*p){**

**char v=p->arr[p->t];**

**p->t--;**

**return v;**

**}**

**int top(struct stack\*p){**

**return p->t;**

**}**

**int main(void) {**

**struct stack\*p=(struct stack\*)malloc(sizeof(struct stack\*));**

**p->t=-1;**

**p->s=5;**

**p->arr=(int \*)malloc(p->s\*sizeof(int ));**

**int i,ans;**

**char postfix[100];**

**char ele;**

**scanf("%s",postfix);**

**i=0;**

**while(i<strlen(postfix)){**

**ele=postfix[i];**

**if(isdigit(ele)){**

**push(p,ele-'0');**

**}else{**

**int num1=pop(p);**

**int num2=pop(p);**

**switch(ele){**

**case '+': ans =num2+num1;**

**break;**

**case '-': ans =num2-num1;**

**break;**

**case '\*': ans =num2\*num1;**

**break;**

**case '/': ans =num2/num1;**

**break;**

**}**

**push(p,ans);**

**}**

**i++;**

**}**

**printf("%d",pop(p));**

**return 0;**

**}**

[**https://practice.geeksforgeeks.org/problems/queue-reversal/1**](https://practice.geeksforgeeks.org/problems/queue-reversal/1)

**without using stacks**

**#include <stdio.h>**

**#include<math.h>**

**#include<stdlib.h>**

**#include<ctype.h>**

**#include<string.h>**

**struct queue{**

**int f;**

**int r;**

**int s;**

**int \*arr;**

**};**

**void en(struct queue \*q,int v){**

**q->r++;**

**q->arr[q->r]=v;**

**}**

**void dq(struct queue \*q){**

**int v=q->arr[q->f];**

**q->f++;**

**}**

**void dis(struct queue\*q,int n){**

**int i;**

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

**printf("%d ",q->arr[i]);**

**}**

**void rev(struct queue\*q){**

**int i,j,n=5,t;**

**for(i=0,j=n-1;i<n/2;i++,j--){**

**t=q->arr[i];**

**q->arr[i]=q->arr[j];**

**q->arr[j]=t;**

**}**

**}**

**int main(void) {**

**struct queue q;**

**q.f=-1;**

**q.r=-1;**

**q.s=5;**

**q.arr=(int \*)malloc(q.s\*sizeof(int \*));**

**en(&q,10);**

**en(&q,20);**

**en(&q,30);**

**en(&q,40);**

**en(&q,50);**

**dis(&q,q.s);**

**rev(&q);**

**dis(&q,q.s);**

**return 0;**

**}**

**Using stacks**

**#include <bits/stdc++.h>**

**using namespace::std;**

**void dis(queue<int>&Queue){**

**int i;**

**while(!Queue.empty()){**

**printf("%d ",Queue.front());**

**Queue.pop();**

**}**

**}**

**void rev(queue<int>&Q){**

**stack<int>s;**

**while(!Q.empty()){**

**s.push(Q.front());**

**Q.pop();**

**}**

**while(!s.empty()){**

**Q.push(s.top());**

**s.pop();**

**}**

**}**

**int main() {**

**queue<int>q;**

**q.push(10);**

**q.push(20);**

**q.push(30);**

**q.push(40);**

**q.push(50);**

**q.push(60);**

**rev(q);**

**dis(q);**

**}**

[**https://www.geeksforgeeks.org/queue-using-stacks/**](https://www.geeksforgeeks.org/queue-using-stacks/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**struct Queue{**

**stack<int>s1,s2;**

**void en(int x){**

**while(!s1.empty()){**

**s2.push(s1.top());**

**s1.pop();**

**}**

**s1.push(x);**

**while(!s2.empty()){**

**s1.push(s2.top());**

**s2.pop();**

**}**

**}**

**int dq(){**

**if(s1.empty())**

**return 0;**

**int y=s1.top();**

**s1.pop();**

**return y;**

**}**

**};**

**int main() {**

**Queue q;**

**q.en(10);**

**q.en(20);**

**q.en(30);**

**cout<<q.dq();**

**cout<<q.dq();**

**cout<<q.dq();**

**}**

**Stack using queues**

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**class st{**

**public:**

**queue<int>q1,q2;**

**void push(int x){**

**q2.push(x);**

**while(!q1.empty()){**

**q2.push(q1.front());**

**q1.pop();**

**}**

**queue<int>q=q1;**

**q1=q2;**

**q2=q;**

**}**

**int pop(){**

**q1.pop();**

**}**

**int size(){**

**return q1.size();**

**}**

**};**

**int main() {**

**st s;**

**s.push(10);**

**s.push(20);**

**s.push(30);**

**cout<<s.size()<<endl;**

**s.pop();**

**cout<<s.size()<<endl;**

**s.pop();**

**cout<<s.size()<<endl;**

**s.pop();**

**cout<<s.size()<<endl;**

**}**

[**https://www.geeksforgeeks.org/reversing-first-k-elements-queue/**](https://www.geeksforgeeks.org/reversing-first-k-elements-queue/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**void mod(queue<int>&q,int k){**

**stack<int>s;**

**int i;**

**int n=q.size();**

**for(i=0;i<k;i++){**

**s.push(q.front());**

**q.pop();**

**}**

**while(!s.empty()){**

**q.push(s.top());**

**s.pop();**

**}**

**for(i=0;i<n-k;i++){**

**q.push(q.front());**

**q.pop();**

**}**

**}**

**void print(queue<int>&q){**

**while(!q.empty()){**

**cout<<q.front()<<" ";**

**q.pop();**

**}**

**}**

**int main() {**

**queue<int>q;**

**q.push(10);**

**q.push(20);**

**q.push(30);**

**q.push(40);**

**q.push(50);**

**q.push(60);**

**q.push(70);**

**q.push(80);**

**q.push(90);**

**q.push(100);**

**mod(q,5);**

**print(q);**

**}**

[**https://www.geeksforgeeks.org/design-a-stack-with-find-middle-operation/**](https://www.geeksforgeeks.org/design-a-stack-with-find-middle-operation/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**struct node{**

**struct node\*left;**

**struct node\*right;**

**int data;**

**node(int x){**

**data=x;**

**left=right=NULL;**

**}**

**};**

**struct node\*dummy=new node(-1);**

**struct node\*head=dummy;**

**struct node\*mid=dummy;**

**int c=0;**

**void push(int data){**

**struct node\*cur=new node(data);**

**cur->left=NULL;**

**cur->right=head;**

**c++;**

**head->left=cur;**

**head=cur;**

**if(c==1){**

**mid=cur;**

**}else if(c%2==0)**

**mid=mid->left;**

**}**

**int pop(){**

**if(c==0){**

**cout<<"stack is empty\n";**

**return -1;**

**}**

**struct node\*temp=head;**

**int x=head->data;**

**head=head->right;**

**if(head!=NULL)**

**head->left=NULL;**

**c--;**

**if(c&1)**

**mid=mid->right;**

**free(temp);**

**return x;;**

**}**

**int middle(){**

**if(c==0){**

**cout<<"stack is empty\n";**

**return -1;**

**}**

**return mid->data;**

**}**

**void print(){**

**struct node\*p=head;**

**int i;**

**for(i=0;i<c;i++){**

**cout<<p->data<<" ";**

**p=p->right;**

**}**

**}**

**int main() {**

**push(1);**

**push(2);**

**push(3);**

**push(4);push(5);**

**push(6);**

**cout<<middle()<<endl;**

**pop();**

**cout<<middle()<<endl;**

**}**

[**https://www.geeksforgeeks.org/length-of-the-longest-valid-substring/**](https://www.geeksforgeeks.org/length-of-the-longest-valid-substring/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**#include<string>**

**int main() {**

**stack<int>s;**

**string exp="(())";**

**int i,l,k=0;**

**s.push(-1);**

**l=exp.length();**

**for(i=0;i<l;i++){**

**if(exp[i]=='(')**

**s.push('(');**

**else {**

**if (!s.empty())**

**{**

**s.pop();**

**}**

**// Check if this length formed with base of**

**// current valid substring is more than max**

**// so far**

**if (!s.empty())**

**k= max(k, i - s.top());**

**// If stack is empty. push current index as**

**// base for next valid substring (if any)**

**else**

**s.push(i);**

**}**

**}**

**printf("%d\n",k);**

**return 0;**

**}**

**2-08-2022**

[**https://www.geeksforgeeks.org/find-expression-duplicate-parenthesis-not/**](https://www.geeksforgeeks.org/find-expression-duplicate-parenthesis-not/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**#include<string>**

**int main() {**

**int i,n;**

**bool ans=false;**

**stack<int>s;**

**string exp="((a+b))";**

**n=exp.length();**

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

**if(exp[i]=='('){**

**s.push(exp[i]);**

**}else if(exp[i]=='+'||exp[i]=='-'||exp[i]=='\*'||exp[i]=='/' ){**

**s.push(exp[i]);**

**}else if(exp[i]==')'){**

**if(s.top()=='(')**

**ans=true;**

**else if(s.top()=='+'||s.top()=='-'||s.top()=='\*'||s.top()=='/' ){**

**s.pop();**

**}**

**s.pop();**

**}**

**}**

**cout<<ans<<" "<<endl;**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/stack-permutations-check-if-an-array-is-stack-permutation-of-other/**](https://www.geeksforgeeks.org/stack-permutations-check-if-an-array-is-stack-permutation-of-other/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**#include<string>**

**int ans(int a[],int b[],int n){**

**int j=0,i;**

**stack<int>s;**

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

**s.push(a[i]);**

**while(!s.empty()&&s.top()==b[j]){**

**s.pop();**

**j++;**

**}**

**}**

**if(s.empty())**

**return 1;**

**else**

**return 0;**

**}**

**int main() {**

**int a[100]={1,2,3};**

**int b[100]={3,1,2};**

**int n=3;**

**cout<<ans(a,b,n)<<endl;**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/sort-a-stack-using-recursion/**](https://www.geeksforgeeks.org/sort-a-stack-using-recursion/)

**#include <iostream>**

**#include<vector>**

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**int sortinsert(stack<int>&s,int x){**

**if(s.empty()||x>s.top())**

**s.push(x);**

**else{**

**int t=s.top();**

**s.pop();**

**sortinsert(s,x);**

**s.push(t);**

**}**

**}**

**int reverse(stack<int>&s){**

**if(s.size()>0){**

**int x=s.top();**

**s.pop();**

**reverse(s);**

**sortinsert(s,x);**

**}**

**return 0;**

**}**

**void sort(stack<int>&s){**

**reverse(s);**

**}**

**void print(stack<int>&s){**

**while(!s.empty()){**

**cout<<s.top()<<" ";**

**s.pop();**

**}**

**}**

**int main() {**

**stack<int>s;**

**s.push(10);**

**s.push(30);**

**s.push(20);**

**sort(s);**

**print(s);**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/queue-based-approach-for-first-non-repeating-character-in-a-stream/**](https://www.geeksforgeeks.org/queue-based-approach-for-first-non-repeating-character-in-a-stream/)

**#include <iostream>**

**#include<vector>**

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespacestd;**

**int main() {**

**string x="aabc";**

**int i;**

**queue<char>q;**

**int n=x.length();**

**int counts[100]={0};**

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

**q.push(x[i]);**

**counts[x[i]-'a']++;**

**while(!q.empty()){**

**if(counts[q.front()-'a']>1)**

**q.pop();**

**else{**

**cout<<q.front()<<" ";**

**break;**

**}**

**}**

**if(q.empty())**

**cout<<"-1 ";**

**}**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/distance-nearest-cell-1-binary-matrix/**](https://www.geeksforgeeks.org/distance-nearest-cell-1-binary-matrix/)

**#include <iostream>**

**#include<vector>**

**#include <iostream>**

**#include <bits/stdc++.h>**

**#include<algorithm>**

**using namespace::std;**

**int main(){**

**int i,j,k,l,ans[100][100],a[100][100],n=3,m=4;**

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

**for(j=0;j<m;j++){**

**cin>>a[i][j];**

**}**

**}**

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

**for(j=0;j<m;j++){**

**ans[i][j]=10000;**

**}**

**}**

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

**for(j=0;j<m;j++){**

**for(k=0;k<n;k++){**

**for(l=0;l<m;l++){**

**if(a[k][l]==1){**

**ans[i][j]=min(ans[i][j],abs(k-i)+abs(l-j));**

**}**

**}**

**}**

**}**

**}**

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

**for(j=0;j<m;j++){**

**cout<<ans[i][j]<<endl;**

**}**

**}**

**return 0;**

**}**

**3-08-2022**

**Stack using queue**

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**class sta{**

**public:**

**queue<int>q;**

**void push(int x){**

**int s=q.size();**

**int i;**

**q.push(x);**

**for(i=0;i<s;i++){**

**q.push(q.front());**

**q.pop();**

**}**

**}**

**int pop(){**

**int x=q.front();**

**q.pop();**

**return x;**

**}**

**};**

**int main() {**

**sta s;**

**s.push(10);**

**s.push(20);**

**s.push(30);**

**s.push(40);**

**cout<<s.pop();**

**cout<<s.pop();**

**cout<<s.pop();**

**}**

[**https://www.geeksforgeeks.org/minimum-time-required-so-that-all-oranges-become-rotten/**](https://www.geeksforgeeks.org/minimum-time-required-so-that-all-oranges-become-rotten/)

**// C++ program to rot all oranges when u can move**

**// in all the four direction from a rotten orange**

**#include <bits/stdc++.h>**

**using namespace std;**

**const int R = 3;**

**const int C = 5;**

**// Check if i, j is under the array limits of row and column**

**bool issafe(int i, int j)**

**{**

**if (i >= 0 && i < R && j >= 0 && j < C)**

**return true;**

**return false;**

**}**

**int rotOranges(int v[R][C])**

**{**

**bool changed = false;**

**int no = 2;**

**while (true) {**

**for (int i = 0; i < R; i++) {**

**for (int j = 0; j < C; j++) {**

**// Rot all other oranges present at**

**// (i+1, j), (i, j-1), (i, j+1), (i-1, j)**

**if (v[i][j] == no) {**

**if (issafe(i + 1, j) && v[i + 1][j] == 1) {**

**v[i + 1][j] = v[i][j] + 1;**

**changed = true;**

**}**

**if (issafe(i, j + 1) && v[i][j + 1] == 1) {**

**v[i][j + 1] = v[i][j] + 1;**

**changed = true;**

**}**

**if (issafe(i - 1, j) && v[i - 1][j] == 1) {**

**v[i - 1][j] = v[i][j] + 1;**

**changed = true;**

**}**

**if (issafe(i, j - 1) && v[i][j - 1] == 1) {**

**v[i][j - 1] = v[i][j] + 1;**

**changed = true;**

**}**

**}**

**}**

**}**

**// if no rotten orange found it means all**

**// oranges rottened now**

**if (!changed)**

**break;**

**changed = false;**

**no++;**

**}**

**for (int i = 0; i < R; i++) {**

**for (int j = 0; j < C; j++) {**

**// if any orange is found to be**

**// not rotten then ans is not possible**

**if (v[i][j] == 1)**

**return -1;**

**}**

**}**

**// Because initial value for a rotten**

**// orange was 2**

**return no - 2;**

**}**

**// Driver function**

**int main()**

**{**

**int v[R][C] = { { 2, 1, 0, 2, 1 },**

**{ 1, 0, 1, 2, 1 },**

**{ 1, 0, 0, 2, 1 } };**

**cout << "Max time incurred: " << rotOranges(v);**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/find-a-tour-that-visits-all-stations/**](https://www.geeksforgeeks.org/find-a-tour-that-visits-all-stations/)

**#include <iostream>**

**#include <bits/stdc++.h>**

**using namespace::std;**

**#include<algorithm>**

**class pump{**

**public:**

**int petrol;**

**int distance;**

**int t(pump p[],int n){**

**int s=0;**

**int r=0;**

**int e=0;**

**int i;**

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

**e+=(p[i].petrol-p[i].distance);**

**if(e<0){**

**s=i+1;**

**r+=e;**

**e=0;**

**}**

**}**

**if(r+e>=0)return s;**

**return -1;**

**}**

**};**

**int main() {**

**pump arr[] = { { 6, 4 }, { 3, 6 }, { 7, 3 } };**

**pump o;**

**cout<< o.t(arr,3);**

**return 0;**

**}**

[**https://www.geeksforgeeks.org/iterative-tower-of-hanoi/**](https://www.geeksforgeeks.org/iterative-tower-of-hanoi/)

**#include <stdio.h>**

**#include<string.h>**

**#include<ctype.h>**

**char toi(int n,char A,char B,char C);**

**int main(void) {**

**int e;**

**scanf("%d",&e);**

**toi(e,'A','B','C');**

**return 0;**

**}**

**char toi(int n,char beg,char aug,char end){**

**if(n>=1){**

**toi(n-1,beg,end,aug);**

**printf("move %d disk from %c to %c\n",n,beg,end);**

**toi(n-1,aug,beg,end);**

**}**

**}**

[**https://www.geeksforgeeks.org/find-the-maximum-of-minimums-for-every-window-size-in-a-given-array/**](https://www.geeksforgeeks.org/find-the-maximum-of-minimums-for-every-window-size-in-a-given-array/)

**#include <stdio.h>**

**#include<string.h>**

**#include<ctype.h>**

**void print(int a[],int n){**

**int i,j,k,max,min;**

**for(k=1;k<=n;k++){**

**max=0;**

**for(i=0;i<=n-k;i++){**

**min=a[i];**

**for(j=1;j<k;j++){**

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

**min=a[i+j];**

**}**

**if(min>max)**

**max=min;**

**}**

**printf("%d\n",max);**

**}**

**}**

**int main(void) {**

**int i,n,a[100]={10,20,30,50,10,70,30};**

**n=7;**

**print(a,n);**

**}**

**7-08-2022**

[**https://leetcode.com/problems/valid-palindrome/**](https://leetcode.com/problems/valid-palindrome/)

**#include <iostream>**

**#include <iostream>**

**#include<string.h>**

**#include <vector>**

**#include<algorithm>**

**using namespace::std;**

**int main() {**

**string s;**

**cin>>s;**

**int i;**

**string so="";**

**for(i=0;i<s.size();i++){**

**if(isdigit(s[i]))**

**so+=s[i];**

**else if(isalpha(s[i]))**

**so+=tolower(s[i]);**

**}**

**string rev=so;**

**reverse(rev.begin(),rev.end());**

**if(so==rev)**

**cout<<"1";**

**else**

**cout<<"0";**

**return 0;**

**}**

<https://leetcode.com/problems/valid-anagram/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

int main() {

string s1,s2;

cin>>s1>>s2;

int f;

if(s1.size()!=s2.size())

cout<<"0";

int c[100]={0};

int i;

for(i=0;i<s1.size();i++)

c[s1[i]-'a']++;

for(i=0;i<s1.size();i++){

c[s2[i]-'a']--;

if(c[s2[i]-'a']<0){

f=0;

break;

}

else

f=1;

}

cout<<f;

return 0;

}

<https://www.geeksforgeeks.org/remove-consecutive-duplicates-string/>

// Recursive Program to remove consecutive

// duplicates from string S.

#include <bits/stdc++.h>

using namespace std;

// A recursive function that removes

// consecutive duplicates from string S

void removeDuplicates(char\* s)

{

// When string is empty, return

if (s[0] == '\0')

return;

if(s[0]==s[1]){

int i=0;

while(s[i]!='\0'){

s[i]=s[i+1];

i++;

}

removeDuplicates(s);

}

// If the adjacent characters are not same

// Check from S+1 string address

removeDuplicates(s + 1);

}

// Driver Program

int main()

{

char S1[] = "geeksforgeeks";

removeDuplicates(S1);

cout << S1 << endl;

char S2[] = "aabcca";

removeDuplicates(S2);

cout << S2 << endl;

return 0;

}

<https://leetcode.com/problems/longest-common-prefix/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

class sol{

public:

string ans(vector<string>&st){

int c=0;

int m=1000;

int i,j;

string s;

for(i=0;i<st.size();i++){

int l=st[i].length();

if(m>l){

m=l;

s=st[i];

}

}

for(i=0;i<m;i++){

for(j=0;j<st.size();j++){

if(st[j][i]!=s[i])

return s.substr(0,c);

}

c++;

}

return s.substr(0,c);

}

};

int main() {

vector<string>st={"flower","flow","flight"};

sol o;

cout<<o.ans(st);

return 0;

}

<https://www.geeksforgeeks.org/convert-sentence-equivalent-mobile-numeric-keypad-sequence/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

string print(string arr[],string s){

string o="";

int i;

int n=s.length();

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

if(s[i]==' ')

o+="0";

else

o+=arr[s[i]-'A'];

}

return o;

}

int main() {

string str[] = {"2","22","222",

"3","33","333",

"4","44","444",

"5","55","555",

"6","66","666",

"7","77","777","7777",

"8","88","888",

"9","99","999","9999"

};

string input = "GEEKSFORGEEKS";

cout << print(str, input);

return 0;

}

<https://leetcode.com/problems/longest-substring-without-repeating-characters/>

#include <iostream>

using namespace::std;

#include<vector>

#include "bits/stdc++.h"

int main() {

string s;

cin>>s;

vector<int>dict(256,-1);

int i,k=-1,m=0;

for(i=0;s[i]!='\0';i++){

if(dict[s[i]]>k)

k=dict[s[i]];

dict[s[i]]=i;

m=max(m,i-k);

}

cout<<m;

return 0;

}

<https://leetcode.com/problems/longest-repeating-character-replacement/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

int main() {

string s="aaba";

int i,st=0,k=1;

vector<int>c(26,0);

int f=0,r=0;

for(i=0;i<s.size();i++){

c[s[i]-'A']++;

f=max(f,c[s[i]-'A']);

while((i-st+1)-f>k){

c[s[i]-'A']--;

st++;

}

r=max(r,i-st+1);

}

cout<<r;

return 0;

}

<https://leetcode.com/problems/longest-palindromic-substring/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

using namespace::std;

int main() {

string s="babad";

int i,j,d,n=4,m=0;

string ans;

vector<vector<int>>dp(n,vector<int>(n,0));

for(d=0;d<4;d++){

for(i=0,j=i+d;j<n;i++,j++){

if(i==j)

dp[i][j]=1;

else if(d==1){

dp[i][j]=(s[i]==s[j])?2:0;

}else{

if(s[i]==s[j]&&dp[i+1][j-1])

dp[i][j]=dp[i+1][j-1]+2;

}

if(dp[i][j]){

if(j-i+1>m){

m=j-i+1;

ans=s.substr(i,j-i+1);

}

}

}

}

cout<<ans;

return 0;

}

09-09-2022

<https://www.geeksforgeeks.org/transform-one-string-to-another-using-minimum-number-of-given-operation/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main() {

string s1="aba";

string s2="aab";

int sum=0;

int i,j;

for(i=0;i<s1.length();i++){

sum+=s1[i];

sum-=s2[i];

}

if(sum!=0)cout<<"impossible";

int c=0;

i=s1.length();

j=s2.length();

while(i>=0&&j>=0){

if(s1[i]!=s2[j]){

c++;

i--;

}else{

i--;

j--;

}

}

cout<<c;

return 0;

}

<https://www.geeksforgeeks.org/longest-prefix-also-suffix/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

string ans() {

string s="abcab";

int i=1,n=s.length();

int len=0;

while(i<n){

if(s[i]==s[len]){

i++;

len++;

}else{

i=i-len+1;

len=0;

}

}

return len>n/2? s.substr(0,len/2):s.substr(0,len);

}

int main(){

cout<<ans();

return 0;

}

10-08-2022

<https://www.geeksforgeeks.org/zigzag-or-diagonal-traversal-of-matrix/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main(){

int i,j,m=3,n=3,k,a[100][100];

for(i=0;i<m;i++){

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

cin>>a[i][j];

}

for(i=0;i<m+n-1;i++){

for(j=0;j<m;j++){

for(k=0;k<n;k++){

if(j+k==i)

cout<<a[k][j];

}

}

cout<<"\n";

}

return 0;

}

<https://leetcode.com/problems/set-matrix-zeroes/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main(){

int i,j,m=3,n=3,k,a[100][100],l;

for(i=0;i<m;i++){

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

cin>>a[i][j];

}

int x=1,y=1;

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

if(a[0][j]==0)

x=0;

}

for(i=0;i<m;i++){

if(a[i][0]==0)

y=0;

}

for(i=1;i<m;i++){

for(j=1;j<n;j++){

if(a[i][j]==0){

a[i][0]=0;

a[0][j]=0;

}

}

}

for(j=0;j<n;j++){

if(a[0][j]==0){

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

a[i][j]=0;

}

}

for(i=0;i<m;i++){

if(a[i][0]==0){

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

a[i][j]=0;

}

}

if(x==0){

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

a[0][j]=0;

}

if(y==0){

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

a[i][0]=0;

}

for(i=0;i<m;i++){

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

cout<<a[i][j];

cout<<"\n";

}

return 0;

}

<https://leetcode.com/problems/spiral-matrix/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main() {

int i,j,m=3,n=3,a[100][100],k=0,l=0;

for(i=0;i<m;i++){

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

cin>>a[i][j];

}

while(k<m&&l<n){

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

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

k++;

for(i=k;i<m;i++){

cout<<a[i][n-1]<<" ";

}

n--;

}

if(k<m){

for(i=n-1;i>=l;i--)

cout<<a[m-1][i];

m--;

}

if(l<n){

for(i=m-1;i>=k;i--)

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

l++;

}

return 0;

}

<https://leetcode.com/problems/rotate-image/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main() {

int i,j,m=3,n=3,a[100][100],t;

for(i=0;i<m;i++){

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

cin>>a[i][j];

}

for(i=0;i<m;i++){

for(j=i;j<n;j++){

t=0;

t=a[i][j];

a[i][j]=a[j][i];

a[j][i]=t;

}

}

for(i=0;i<m;i++){

for(j=0;j<n/2;j++){

t=a[i][j];

a[i][j]=a[i][n-1-j];

a[i][n-1-j]=t;

}

}

for(i=0;i<m;i++){

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

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

cout<<"\n";

}

return 0;

}

<https://leetcode.com/problems/word-search/>

#include <bits/stdc++.h>

using namespace std;

int row,col;

int dx**[**4**]** = **{**0, 1, 0, -1**}**;

int dy**[**4**]** = **{**1, 0, -1, 0**}**;

bool backtrack**(**int i,int j,vector**<**vector**<**char**>>**& board, string word,unsigned int ind**)**

**{**

if**(**ind**>**=word.size**())** return true;

if**(**i**<**0 || i**>**=row || j**<**0 || j**>**=col || board**[**i**][**j**]**!=word**[**ind**])** return false;

board**[**i**][**j**]**= '#';

char t=board**[**i**][**j**]**;

for**(**int k=0;k**<**4;k++**)**

**{**

if**(**backtrack**(**i+dx**[**k**]**,j+dy**[**k**]**,board,word,ind+1**))**

return true;

**}**

board**[**i**][**j**]** = t;

return false;

**}**

bool exist**(**vector**<**vector**<**char**>>**& board, string word**)**

**{**

row= board.size**()**;

col= board**[**0**]**.size**()**;

for**(**int i=0;i**<**row;i++**)**

for**(**int j=0;j**<**col;j++**)**

if**(**backtrack**(**i,j,board,word,0**))**

return true;

return false;

**}**

int main**()**

**{**

vector**<**vector**<**char**>>** board= **{**

**{**'A','B','C','E'**}**,

**{**'S','F','C','S'**}**,

**{**'A','D','E','E'**}**

**}**;

string word = "ABCCED";

if**(**exist**(**board,word**))**

cout**<<** "true" ;

else

cout**<<** "false" ;

return 0;

**}**

11-08-2022

<https://www.geeksforgeeks.org/largest-rectangle-under-histogram/>

#include <iostream>

#include <iostream>

#include<string.h>

#include <vector>

#include<algorithm>

#include <bits/stdc++.h>

using namespace::std;

int main() {

vector<int>h={6,2,5,4,5,1,6};

int i,n=h.size();

stack<int>s;

vector<int>left(n,0);

vector<int>right(n,0);

int m=0;

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

if(s.empty()){

left[i]=0;

s.push(i);

}else if(h[s.top()]>=h[i]){

while(!s.empty()&&h[s.top()]>=h[i])

s.pop();

if(s.empty())left[i]=0;

else left[i]=s.top()+1;

s.push(i);

}else{

left[i]=s.top()+1;

s.push(i);

}

}

while(!s.empty())

s.pop();

int j;

for(j=n-1;j>=0;j--){

if(s.empty()){

right[j]=n-1;

s.push(j);

}else if(h[s.top()]>=h[j]){

while(!s.empty()&&h[s.top()]>=h[j])

s.pop();

if(s.empty())right[j]=n-1;

else right[j]=s.top()-1;

s.push(j);

}else{

right[j]=s.top()-1;

s.push(j);

}

}

int k;

for(k=0;k<n;k++){

if((right[k]-left[k]+1)\*h[k]>=m)

m=(right[k]-left[k]+1)\*h[k];

}

cout<<m;

return 0;

}

12-08-2022

<https://www.geeksforgeeks.org/maximum-sum-rectangle-in-a-2d-matrix-dp-27/#:~:text=Given%20a%202D%20array%2C%20find,Contiguous%20Subarray%20for%201D%20array>.

#include <iostream>

using namespace::std;

#include<vector>

#include<algorithm>

int kadane(vector<int>v,int c){

int ma=INT\_MIN;

int u=0;

int i;

for(i=0;i<c;i++){

u+=v[i];

if(u>ma)ma=u;

if(u<0)u=0;

}

return ma;

}

int main() {

int r,c;

cin>>r>>c;

vector<vector<int>>mat(r,vector<int>(c));

int i,j;

for(i=0;i<r;i++){

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

cin>>mat[i][j];

}

int ma=INT\_MIN;

int col;

for(i=0;i<r;i++){

vector<int>ans(c);

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

for(col=0;col<c;col++)

ans[col]+=mat[j][col];

ma=max(ma,kadane(ans,c));

}

}

cout<<ma<<endl;

}

<https://www.geeksforgeeks.org/create-a-matrix-with-alternating-rectangles-of-0-and-x/>

#include <iostream>

using namespace::std;

#include<vector>

#include<algorithm>

void ans(int m,int n) {

int r=m,c=n;

char a[m][n];

int i,j;

char x='X';

int k=0,l=0;

while(k<m&&l<n){

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

a[k][i]=x;

k++;

for(i=k;i<m;i++)

a[i][n-1]=x;

n--;

if(k<m){

for(i=n-1;i>=l;i--)

a[m-1][i]=x;

m--;

}

if(l<n){

for(i=m-1;i>=k;i--)

a[i][l]=x;

l++;

}

x = (x == '0')? 'X': '0';

}

for(i=0;i<r;i++){

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

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

cout<<"\n";

}

}

int main(){

ans(3,3);

return 0;

}

<https://www.geeksforgeeks.org/find-common-element-rows-row-wise-sorted-matrix/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

// Specify number of rows and columns

#define M 4

#define N 5

// Returns common element in all rows of mat[M][N]. If there is no

// common element, then -1 is returned

int findCommon(int mat[M][N])

{

// A hash map to store count of elements

unordered\_map<int, int> cnt;

int i, j;

for (i = 0; i < M; i++) {

// Increment the count of first

// element of the row

cnt[mat[i][0]]++;

// Starting from the second element

// of the current row

for (j = 1; j < N; j++) {

// If current element is different from

// the previous element i.e. it is appearing

// for the first time in the current row

if (mat[i][j] != mat[i][j - 1])

cnt[mat[i][j]]++;

}

}

// Find element having count equal to number of rows

for (auto ele : cnt) {

if (ele.second == M)

return ele.first;

}

// No such element found

return -1;

}

// Driver Code

int main()

{

int mat[M][N] = {

{ 1, 2, 3, 4, 5 },

{ 2, 4, 5, 8, 10 },

{ 3, 5, 7, 9, 11 },

{ 1, 3, 5, 7, 9 },

};

int result = findCommon(mat);

if (result == -1)

cout << "No common element";

else

cout << "Common element is " << result;

return 0;

}

<https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint/>

#include <iostream>

using namespace::std;

#include<vector>

#include<algorithm>

#include <bits/stdc++.h>

const int m=8;

const int n=8;

void dfs(int i,int j,int vis[m][n],int screen[m][n],int c,int m,int n,int o){

if(i<0||j<0||i>=m||j>=n) return ;

if(vis[i][j]||screen[i][j]!=o)return ;

vis[i][j]=1;

screen[i][j]=c;

dfs(i+1,j,vis,screen,c,m,n,o);

dfs(i-1,j,vis,screen,c,m,n,o);

dfs(i,j+1,vis,screen,c,m,n,o);

dfs(i,j-1,vis,screen,c,m,n,o);

}

int floodFill(int screen[m][n],int x,int y,int c){

int vis[m][n];

int o=screen[x][y];

dfs(x,y,vis,screen,c,m,n,o);

}

int main(){

int screen[m][n] = {{1, 1, 1, 1, 1, 1, 1, 1},

{1, 1, 1, 1, 1, 1, 0, 0},

{1, 0, 0, 1, 1, 0, 1, 1},

{1, 2, 2, 2, 2, 0, 1, 0},

{1, 1, 1, 2, 2, 0, 1, 0},

{1, 1, 1, 2, 2, 2, 2, 0},

{1, 1, 1, 1, 1, 2, 1, 1},

{1, 1, 1, 1, 1, 2, 2, 1},

};

int x = 4, y = 4, newC = 3;

floodFill(screen, x, y, newC);

cout << "Updated screen after call to floodFill: \n";

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

{

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

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

cout << endl;

}

return 0;

}

<https://www.geeksforgeeks.org/given-matrix-o-x-replace-o-x-surrounded-x/>

// A C++ program to replace all 'O's with 'X''s if surrounded by 'X'

#include<iostream>

using namespace std;

// Size of given matrix is M X N

#define M 6

#define N 6

// A recursive function to replace previous value 'prevV' at '(x, y)'

// and all surrounding values of (x, y) with new value 'newV'.

void floodFillUtil(char mat[][N], int x, int y, char prevV, char newV)

{

// Base cases

if (x < 0 || x >= M || y < 0 || y >= N)

return;

if (mat[x][y] != prevV)

return;

// Replace the color at (x, y)

mat[x][y] = newV;

// Recur for north, east, south and west

floodFillUtil(mat, x+1, y, prevV, newV);

floodFillUtil(mat, x-1, y, prevV, newV);

floodFillUtil(mat, x, y+1, prevV, newV);

floodFillUtil(mat, x, y-1, prevV, newV);

}

// Returns size of maximum size subsquare matrix

// surrounded by 'X'

int replaceSurrounded(char mat[][N])

{

// Step 1: Replace all 'O' with '-'

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

for (int j=0; j<N; j++)

if (mat[i][j] == 'O')

mat[i][j] = '-';

// Call floodFill for all '-' lying on edges

for (int i=0; i<M; i++) // Left side

if (mat[i][0] == '-')

floodFillUtil(mat, i, 0, '-', 'O');

for (int i=0; i<M; i++) // Right side

if (mat[i][N-1] == '-')

floodFillUtil(mat, i, N-1, '-', 'O');

for (int i=0; i<N; i++) // Top side

if (mat[0][i] == '-')

floodFillUtil(mat, 0, i, '-', 'O');

for (int i=0; i<N; i++) // Bottom side

if (mat[M-1][i] == '-')

floodFillUtil(mat, M-1, i, '-', 'O');

// Step 3: Replace all '-' with 'X'

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

for (int j=0; j<N; j++)

if (mat[i][j] == '-')

mat[i][j] = 'X';

}

// Driver program to test above function

int main()

{

char mat[][N] = {{'X', 'O', 'X', 'O', 'X', 'X'},

{'X', 'O', 'X', 'X', 'O', 'X'},

{'X', 'X', 'X', 'O', 'X', 'X'},

{'O', 'X', 'X', 'X', 'X', 'X'},

{'X', 'X', 'X', 'O', 'X', 'O'},

{'O', 'O', 'X', 'O', 'O', 'O'},

};

replaceSurrounded(mat);

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

{

for (int j=0; j<N; j++)

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

cout << endl;

}

return 0;

}

13-08-2022

<https://leetcode.com/problems/maximum-depth-of-binary-tree/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=new node;

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\*ins(struct node\*my,int x){

if(my==NULL)

return newnode(x);

else{

if(x<my->data)

my->left=ins(my->left,x);

else if(x>my->data);

my->right=ins(my->right,x);

}

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

int maxDepth(node \* root)

{

// Root being null means tree doesn't exist.

if (root == NULL)

return 0;

// Get the depth of the left and right subtree

// using recursion.

int leftDepth = maxDepth(root->left);

int rightDepth = maxDepth(root->right);

// Choose the larger one and add the root to it.

if (leftDepth > rightDepth)

return leftDepth + 1;

else

return rightDepth + 1;

}

int height(node\*my){

if(my==NULL)return 0;

else{

int l=height(my->left);

int r=height(my->right);

if(l>r)

return l+1;

else

return r+1;

}

}

void print(struct node\*my,int h){

if(my==NULL)return;

if(h==1)cout<<my->data<<" ";

else{

print(my->left,h-1);

print(my->right,h-1);

}

}

int level(node\*my){

int i,h=height(my);

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

print(my,i);

}

int main(){

struct node\* root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

cout<<maxDepth(root);

return 0;

}

<https://www.geeksforgeeks.org/level-order-tree-traversal/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=new node;

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\*ins(struct node\*my,int x){

if(my==NULL)

return newnode(x);

else{

if(x<my->data)

my->left=ins(my->left,x);

else if(x>my->data);

my->right=ins(my->right,x);

}

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

int maxDepth(node \* root)

{

// Root being null means tree doesn't exist.

if (root == NULL)

return 0;

// Get the depth of the left and right subtree

// using recursion.

int leftDepth = maxDepth(root->left);

int rightDepth = maxDepth(root->right);

// Choose the larger one and add the root to it.

if (leftDepth > rightDepth)

return leftDepth + 1;

else

return rightDepth + 1;

}

int heigth(node\*my){

if(my==NULL)

return 0;

else{

int l=heigth(my->left);

int r=heigth(my->right);

if(l>r)

return l+1;

else

return r+1;

}

}

void print(node\*my,int i){

if(my==NULL)return ;

if(i==1)cout<<my->data<<" ";

else{

print(my->left,i-1);

print(my->right,i-1);

}

}

int level(node\*my){

int i;

int h=heigth(my);

for(i=1;i<=h;i++){

print(my,i);

}

}

int main(){

struct node\* root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

level(root);

return 0;

}

<https://www.geeksforgeeks.org/check-if-a-binary-tree-is-subtree-of-another-binary-tree/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\*ins(struct node\*my,int v){

if(my==NULL)

return newnode(v);

else{

if(v<my->data)

my->left=ins(my->left,v);

else if(v>my->data)

my->right=ins(my->right,v);

}

}

bool ide(struct node\*my1,struct node\*my2){

if(my1==NULL&&my2==NULL)

return 1;

if(my1==NULL||my2==NULL)

return 0;

return (my1->data==my2->data&&ide(my1->left,my2->left)&&ide(my1->right,my2->right));

}

bool ise(struct node\*my1,struct node\*my2){

if(my1==NULL)

return 0;

if(my2==NULL)

return 1;

if(ide(my1,my2)) return 1;

return ise(my1->left,my2)||ise(my1->right,my2);

}

int main(){

node \*T = newnode(26);

T->right = newnode(3);

T->right->right = newnode(3);

T->left = newnode(10);

T->left->left = newnode(4);

T->left->left->right = newnode(30);

T->left->right = newnode(6);

node \*S = newnode(10);

S->right = newnode(6);

S->left = newnode(4);

S->left->right = newnode(30);

if (ise(T, S))

cout << "Tree 2 is subtree of Tree 1";

else

cout << "Tree 2 is not a subtree of Tree 1";

return 0;

}

<https://www.geeksforgeeks.org/write-an-efficient-c-function-to-convert-a-tree-into-its-mirror-tree/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

struct node\*ins(struct node\*my,int v){

if(my==NULL)

return newnode(v);

else{

if(v<my->data)

my->left=ins(my->left,v);

else if(v>my->data)

my->right=ins(my->right,v);

}

}

int ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void mirror(struct node\*my){

if(my==NULL)return;

else{

mirror(my->left);

mirror(my->right);

struct node\*t=my->left;

my->left=my->right;

my->right=t;

}

}

int main(){

struct node \*root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

/\* Print inorder traversal of the input tree \*/

cout << "Inorder traversal of the constructed"

<< " tree is" << endl;

ino(root);

/\* Convert tree to its mirror \*/

mirror(root);

/\* Print inorder traversal of the mirror tree \*/

cout << "\nInorder traversal of the mirror tree"

<< " is \n";

ino(root);

return 0;

}

<https://www.geeksforgeeks.org/print-left-view-binary-tree/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

void leftview(struct node\*my){

queue<struct node\*>q;

int i;

q.push(my);

while(!q.empty()){

int n=q.size();

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

struct node\*t=q.front();

q.pop();

if(i==1)

cout<<t->data<<" ";

if(t->left!=NULL)

q.push(t->left);

if(t->right!=NULL)

q.push(t->right);

}

}

}

int main(){

struct node \*root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

leftview(root);

return 0;

}

<https://practice.geeksforgeeks.org/problems/right-view-of-binary-tree/1>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

void rightview(struct node\*my){

queue<struct node\*>q;

int i;

q.push(my);

while(!q.empty()){

int n=q.size();

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

struct node\*t=q.front();

q.pop();

if(i==n-1)

cout<<t->data<<" ";

if(t->left!=NULL)

q.push(t->left);

if(t->right!=NULL)

q.push(t->right);

}

}

}

int main(){

struct node \*root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

rightview(root);

return 0;

}

<https://www.geeksforgeeks.org/zigzag-tree-traversal/>

// C++ implementation of the approach

#include <bits/stdc++.h>

using namespace std;

struct node{

int data;

struct node\*left;

struct node\*right;

};

struct node\*newnode(int x){

struct node\*temp=(struct node\*)malloc(sizeof(struct node\*));

temp->data=x;

temp->left=NULL;

temp->right=NULL;

return temp;

}

void zig(struct node\*my){

if(!my)

return;

stack<struct node\*>cl;

stack<struct node\*>nl;

cl.push(my);

bool ltr=true;

while(!cl.empty()){

struct node\*t=cl.top();

cl.pop();

if(t){

cout<<t->data<<" ";

if(ltr){

if(t->left)

nl.push(t->left);

if(t->right)

nl.push(t->right);

}else{

if(t->left)

nl.push(t->left);

if(t->right)

nl.push(t->right);

}

}

if(cl.empty()){

ltr=!ltr;

swap(cl,nl);

}

}

}

int main(){

struct node \*root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

zig(root);

return 0;

}

14-08-2022

<https://www.geeksforgeeks.org/check-leaves-level/>

// C++ program to check if all leaves

// are at same level

#include <bits/stdc++.h>

using namespace std;

// A binary tree node

struct Node

{

int data;

struct Node \*left, \*right;

};

// A utility function to allocate

// a new tree node

struct Node\* newNode(int data)

{

struct Node\* node = (struct Node\*) malloc(sizeof(struct Node));

node->data = data;

node->left = node->right = NULL;

return node;

}

bool check2(struct Node\*my,int level,int \*leaflevel){

if(my==NULL)return 1;

if(my->left==NULL&&my->right==NULL){

if(leaflevel==0){

\*leaflevel=level;

return 1;

return level=\*leaflevel;

}

}

return check2(my->left,level+1,leaflevel)&&check2(my->right,level+1,leaflevel);

}

bool check(struct Node \*root)

{

int level = 0, leaflevel = 0;

return check2(root, level, &leaflevel);

}

int main()

{

// Let us create tree shown in third example

struct Node \*root = newNode(12);

root->left = newNode(5);

root->left->left = newNode(3);

root->left->right = newNode(9);

root->left->left->left = newNode(1);

root->left->right->left = newNode(1);

if (check(root))

cout << "Leaves are at same level\n";

else

cout << "Leaves are not at same level\n";

getchar();

return 0;

}

// This code is contributed

// by Akanksha Rai

<https://www.geeksforgeeks.org/how-to-determine-if-a-binary-tree-is-balanced/>

// C++ program to check if all leaves

// are at same level

#include <bits/stdc++.h>

using namespace std;

// A binary tree node

struct node

{

int data;

struct node \*left, \*right;

};

// A utility function to allocate

// a new tree node

struct node\* newnode(int data)

{

struct node\* node = (struct node\*) malloc(sizeof(struct node));

node->data = data;

node->left = node->right = NULL;

return node;

}

int height(struct node\*my){

if(my==NULL)return 0;

int lh=height(my->left);

int rh=height(my->right);

if(lh>rh)

return lh+1;

else

return rh+1;

}

int bal(struct node\*my){

if(my==NULL)return 1;

int lh=height(my->left);

int rh=height(my->right);

if((abs(lh-rh)<=1&&bal(my->left)&&bal(my->right)))

return 1;

return 0;

}

int main()

{

node\* root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

if (bal(root))

cout << "Tree is balanced";

else

cout << "Tree is not balanced";

return 0;

}

// This code is contributed

// by Akanksha Rai

<https://www.geeksforgeeks.org/convert-a-given-tree-to-sum-tree/>

// C++ program to convert a tree into its sum tree

#include <bits/stdc++.h>

using namespace std;

/\* A tree node structure \*/

class node

{

public:

int data;

node \*left;

node \*right;

};

// Convert a given tree to a tree where

// every node contains sum of values of

// nodes in left and right subtrees in the original tree

int toSumTree(struct node\*my){

if(my==NULL)return 0;

int o=my->data;

my->data=toSumTree(my->left)+toSumTree(my->right);

return o+my->data;

}

void printInorder(node\* Node)

{

if (Node == NULL)

return;

printInorder(Node->left);

cout<<" "<<Node->data;

printInorder(Node->right);

}

/\* Utility function to create a new Binary Tree node \*/

node\* newNode(int data)

{

node \*temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return temp;

}

/\* Driver code \*/

int main()

{

node \*root = NULL;

int x;

/\* Constructing tree given in the above figure \*/

root = newNode(10);

root->left = newNode(-2);

root->right = newNode(6);

root->left->left = newNode(8);

root->left->right = newNode(-4);

root->right->left = newNode(7);

root->right->right = newNode(5);

toSumTree(root);

// Print inorder traversal of the converted

// tree to test result of toSumTree()

cout<<"Inorder Traversal of the resultant tree is: \n";

printInorder(root);

return 0;

}

// This code is contributed by rathbhupendra

<https://www.geeksforgeeks.org/tree-isomorphism-problem/>

// A C++ program to check if two given trees are isomorphic

#include <iostream>

using namespace std;

/\* A binary tree node has data, pointer to left and right children \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

/\* Given a binary tree, print its nodes in reverse level order \*/

bool isIsomorphic(node\* n1, node \*n2)

{

// Both roots are NULL, trees isomorphic by definition

if (n1 == NULL && n2 == NULL)

return true;

// Exactly one of the n1 and n2 is NULL, trees not isomorphic

if (n1 == NULL || n2 == NULL)

return false;

if (n1->data != n2->data)

return false;

// There are two possible cases for n1 and n2 to be isomorphic

// Case 1: The subtrees rooted at these nodes have NOT been "Flipped".

// Both of these subtrees have to be isomorphic, hence the &&

// Case 2: The subtrees rooted at these nodes have been "Flipped"

return

(isIsomorphic(n1->left,n2->left) && isIsomorphic(n1->right,n2->right))||

(isIsomorphic(n1->left,n2->right) && isIsomorphic(n1->right,n2->left));

}

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

node\* newNode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

/\* Driver program to test above functions\*/

int main()

{

// Let us create trees shown in above diagram

struct node \*n1 = newNode(1);

n1->left = newNode(2);

n1->right = newNode(3);

n1->left->left = newNode(4);

n1->left->right = newNode(5);

n1->right->left = newNode(6);

n1->left->right->left = newNode(7);

n1->left->right->right = newNode(8);

struct node \*n2 = newNode(1);

n2->left = newNode(3);

n2->right = newNode(2);

n2->right->left = newNode(4);

n2->right->right = newNode(5);

n2->left->right = newNode(6);

n2->right->right->left = newNode(8);

n2->right->right->right = newNode(7);

if (isIsomorphic(n1, n2) == true)

cout << "Yes";

else

cout << "No";

return 0;

}

<https://www.geeksforgeeks.org/construct-tree-from-given-inorder-and-preorder-traversal/>

#include <bits/stdc++.h>

using namespace std;

// A binary tree node

struct node

{

int data;

struct node \*left, \*right;

};

// A utility function to allocate

// a new tree node

struct node\* newnode(int data)

{

struct node\* node = (struct node\*) malloc(sizeof(struct node));

node->data = data;

node->left = node->right = NULL;

return node;

}

int id=0;

unordered\_map<int,int>m;

struct node\*solve(int pre[],int in[],int lb,int ub){

if(lb>ub)return NULL;

struct node\*res=newnode(pre[id++]);

if(lb==ub)return res;

int mid=m[res->data];

res->left=solve(pre,in,lb,mid-1);

res->right=solve(pre,in,mid+1,ub);

return res;

}

struct node\*buld(int in[],int pre[],int n){

id=0;

m.clear();

int i;

for(i=0;i<n;i++)m[in[i]]=i;

struct node\*root=solve(pre,in,0,n-1);

return root;

}

struct node\*ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

int main()

{

// Let us create tree shown in third example

int in[]={3,1,4,0,5,2};

int pre[]={0,1,3,4,2,5};

struct node\*my=buld(in,pre,6);

ino(my);

return 0;

}

<https://www.geeksforgeeks.org/diameter-of-a-binary-tree/>

// A C++ program to check if two given trees are isomorphic

#include <iostream>

using namespace std;

/\* A binary tree node has data, pointer to left and right children \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

int height(struct node\*my){

if(my==NULL)

return 0;

int lh=height(my->left);

int rh=height(my->right);

if(lh>rh)return lh+1;

else

return rh+1;

}

int dia(struct node\*my){

if(my==NULL)

return 0;

int ld=dia(my->left);

int rd=dia(my->right);

return max(height(my->left)+height(my->right)+1,max(dia(my->left),dia(my->right)));

}

int main()

{

struct node\* root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->left->right = newnode(5);

cout<<diameter(root);

return 0;

}

<https://www.geeksforgeeks.org/print-k-sum-paths-binary-tree/>

// A C++ program to check if two given trees are isomorphic

#include <iostream>

using namespace std;

#include<vector>

/\* A binary tree node has data, pointer to left and right children \*/

void fun(struct node\*my,vector<int>&path,int k);

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void fun(struct node\*my,vector<int>&path,int k){

if(my==NULL)return;

path.push\_back(my->data);

fun(my->left,path,k);

fun(my->right,path,k);

int f=0,j,m;

for(j=path.size();j>=1;j--){

f+=path[j];

if(f==k){

for(m=j;m<path.size();m++){

cout<<path[m]<<" ";

}

cout<<"\n";

}

}

path.pop\_back();

}

int main()

{

node\* root = newnode(1);

root->left = newnode(3);

root->left->left = newnode(2);

root->left->right = newnode(1);

root->left->right->left = newnode(1);

root->right = newnode(-1);

root->right->left = newnode(4);

root->right->left->left = newnode(1);

root->right->left->right = newnode(2);

root->right->right = newnode(5);

root->right->right->right = newnode(2);

int k = 5;

vector<int>path;

fun(root,path, k);

return 0;

}

<https://www.geeksforgeeks.org/convert-given-binary-tree-doubly-linked-list-set-3/>

#include <iostream>

using namespace std;

#include<vector>

/\* A binary tree node has data, pointer to left and right children \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void solve(struct node\*my,struct node\* &head,struct node\* &p,int &f){

if(my==NULL)return;

solve(my->left,head,p,f);

if(f==0){

f=1;

head=my;

p=my;

}else{

p->right=my;

p->right->left=p;

p=p->right;

}

solve(my->right,head,p,f);

}

struct node\*ans(struct node\*my){

struct node\*head=NULL;

struct node\*p=NULL;

int f=0;

solve(my,head,p,f);

return head;

}

void dis(struct node\*\*head){

struct node\*p=\*head;

while(p->right!=NULL){

cout<<p->data<<" ";

p=p->right;

}

}

int main()

{

node\* root = newnode(1);

root->left = newnode(3);

root->left->left = newnode(2);

root->left->right = newnode(1);

root->left->right->left = newnode(1);

root->right = newnode(-1);

root->right->left = newnode(4);

root->right->left->left = newnode(1);

root->right->left->right = newnode(2);

root->right->right = newnode(5);

root->right->right->right = newnode(2);

struct node\*my=ans(root);

dis(&my);

return 0;

}

<https://www.geeksforgeeks.org/find-maximum-path-sum-in-a-binary-tree/>

#include <iostream>

using namespace std;

#include<vector>

/\* A binary tree node has data, pointer to left and right children \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void solve(struct node\*my,struct node\*head,struct node\*p,int &f){

if(my==NULL)return;

solve(my->left,head,p,f);

if(f==0){

f=1;

head=my;

p=my;

}else{

p->right=my;

p->right->left=p;

p=p->right;

}

solve(my->right,head,p,f);

}

struct node\*ans(struct node\*my){

struct node\*head=NULL;

struct node\*p=NULL;

int f=0;

solve(my,head,p,f);

return head;

}

void dis(struct node\*\*head){

struct node\*p=\*head;

while(p->right!=NULL){

cout<<p->data<<" ";

p=p->right;

}

}

int fmp(struct node\*my,int& ans){

if(my==NULL)return 0;

int l=fmp(my->left,ans);

int r=fmp(my->right,ans);

if(l<0)l=0;

if(r<0)r=0;

ans=max(ans,l+r+my->data);

return max(l,r)+my->data;

}

int findMaxSum(struct node\*my){

int ans=INT8\_MIN;

fmp(my,ans);

return ans;

}

int main()

{

struct node \*root = newnode(10);

root->left = newnode(2);

root->right = newnode(10);

root->left->left = newnode(20);

root->left->right = newnode(1);

root->right->right = newnode(-25);

root->right->right->left = newnode(3);

root->right->right->right = newnode(4);

cout << "Max path sum is " << findMaxSum(root);

return 0;

}

15-08-2022

<https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/>

#include <iostream>

using namespace::std;

#include<algorithm>

#include<vector>

int main() {

int insc=-1,ind;

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

int n=a.size();

int i;

int st=0,end,s=0,sum=0;

int max=0;

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

sum+=a[i];

if(sum>max){

max=sum;

st=s;

end=i;

}

if(sum<0){

sum=0;

s=i+1;

}

}

cout<<max<<" "<<st<<" "<<end<<" ";

}

Sub array of another array

#include <iostream>

using namespace::std;

#include<algorithm>

#include<vector>

int fun(vector<int>a,vector<int>b){

int m=a.size();

int n=b.size();

int i,j;

for(j=0;j<n;j++){

for(i=0;i<m;i++){

if(b[j]==a[i])

break;

}

if(j==n)

return 0;

}

return 1;

}

int main() {

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

vector<int>b={1,2,3};

cout<<fun(a,b);

}

16-08-2022

**NODES AT DISTANCE K**

#include <iostream>

using namespace std;

#include<vector>

/\* A binary tree node has data, pointer to left and right children \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void dis(struct node\*\*head){

struct node\*p=\*head;

while(p->right!=NULL){

cout<<p->data<<" ";

p=p->right;

}

}

void ps(struct node\*my,int k){

if(my==NULL||k<0)

return;

if(k==0){

cout<<my->data<<" ";

return;

}

ps(my->left,k-1);

ps(my->right,k-1);

}

int pa(struct node\*my,struct node\*tar,int k){

if(my==NULL)return -1;

if(my==tar){

ps(my,k);

return 0;

}

int dl=pa(my->left,tar,k);

if(dl!=-1){

if(dl+1==k)

cout<<my->data<<" ";

else

ps(my->right,k-dl-2);

return 1+dl;

}

int dr=pa(my->right,tar,k);

if(dr!=-1){

if(dr+1==k)

cout<<my->data<<" ";

else

ps(my->left,k-dr-2);

return 1+dr;

}

return -1;

}

int main()

{

struct node \*root = newnode(10);

root->left = newnode(2);

root->right = newnode(10);

root->left->left = newnode(20);

root->left->right = newnode(1);

root->right->right = newnode(-25);

root->right->right->left = newnode(3);

root->right->right->right = newnode(4);

pa(root,root->left->left,1);

return 0;

}

<https://www.geeksforgeeks.org/find-duplicate-subtrees/>

#include <iostream>

using namespace std;

#include<vector>

#include <bits/stdc++.h>

#include<string>

/\* A binary tree node has data, pointer to left and right children \*/

#include<string>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void dis(struct node\*\*head){

struct node\*p=\*head;

while(p->right!=NULL){

cout<<p->data<<" ";

p=p->right;

}

}

string ans(struct node\*my,unordered\_map<string,int>&m){

if(my==NULL)

return "";

string str="(";

str+=ans(my->left,m);

str+=to\_string(my->data);

str+=ans(my->right,m);

str+=")";

if(m[str]==1)

cout<<my->data<<" ";

m[str]++;

return str;

};

void print(struct node\*my){

unordered\_map<string,int>m;

ans(my,m);

}

int main()

{

node\* root = NULL;

root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->right->left = newnode(2);

root->right->left->left = newnode(4);

root->right->right = newnode(4);

print(root);

return 0;

}

<https://practice.geeksforgeeks.org/problems/min-distance-between-two-given-nodes-of-a-binary-tree/1>

#include <iostream>

using namespace std;

#include<vector>

#include <bits/stdc++.h>

#include<string>

/\* A binary tree node has data, pointer to left and right children \*/

#include<string>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

struct node\*lca(struct node\*my,int n1,int n2){

if(my==NULL)

return NULL;

if(my->data==n1||my->data==n2)

return my;

struct node\*l=lca(my->left,n1,n2);

struct node\*r=lca(my->right,n1,n2);

if(l!=NULL&&r!=NULL)

return my;

if(l==NULL&&r==NULL)

return NULL;

if(l!=NULL)

return lca(my->left,n1,n2);

return lca(my->right,n1,n2);

}

int fd(struct node\*my,int k,int dis){

if(my==NULL)

return -1;

if(my->data==k)

return dis;

int l=fd(my->left,k,dis+1);

if(l!=-1)

return l;

return fd(my->right,k,dis+1);

}

int dis(struct node\*my,int n1,int n2){

struct node\*l=lca(my,n1,n2);

int d1=fd(l,n1,0);

int d2=fd(l,n2,0);

return d1+d2;

}

int main()

{

node\* root = NULL;

root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->right->left = newnode(2);

root->right->left->left = newnode(4);

root->right->right = newnode(4);

cout<<dis(my,1,2);

return 0;

}

17-08-2022

<https://www.geeksforgeeks.org/minimum-number-swaps-required-sort-array/>

#include <iostream>

using namespace std;

#include<vector>

#include <bits/stdc++.h>

#include<string>

/\* A binary tree node has data, pointer to left and right children \*/

#include<string>

int minSwaps(int arr[],int n){

int i,s=0;

vector<pair<int,int>>v;

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

v.push\_back({arr[i],i});

sort(v.begin(),v.end());

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

pair<int,int>p=v[i];

int val=p.first;

int ind=p.second;

if(i!=ind){

s++;

swap(v[i],v[ind]);

i--;

}

}

return s;

}

int main()

{

int arr[] = {1, 5, 4, 3, 2};

int n = (sizeof(arr) / sizeof(int));

cout << minSwaps(arr, n);

return 0;

}

Conversion of binary tree to binary search tree

#include <iostream>

using namespace std;

#include<vector>

#include <bits/stdc++.h>

#include<string>

/\* A binary tree node has data, pointer to left and right children \*/

#include<string>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void ino(struct node\*my){

if(my!=NULL){

ino(my->left);

cout<<my->data<<" ";

ino(my->right);

}

}

void ino\_tree(struct node\*my,vector<int>&v){

if(my==NULL)

return;

ino\_tree(my->left,v);

v.push\_back(my->data);

ino\_tree(my->right,v);

}

void ino\_bst(struct node\*my,vector<int>v,int &i){

if(my==NULL)return;

ino\_bst(my->left,v,i);

my->data=v[i];

i++;

ino\_bst(my->right,v,i);

}

struct node\*conv(struct node\*my){

vector<int>v;

ino\_tree(my,v);

sort(v.begin(),v.end());

int i=0;

ino\_bst(my,v,i);

return my;

}

int main()

{

node\* root = NULL;

root = newnode(1);

root->left = newnode(2);

root->right = newnode(3);

root->left->left = newnode(4);

root->right->left = newnode(2);

root->right->left->left = newnode(4);

root->right->right = newnode(4);

ino(root);

struct node\*my=conv(root);

cout<<"\n";

ino(root);

return 0;

}

<https://www.geeksforgeeks.org/construct-binary-tree-string-bracket-representation/>

#include <iostream>

using namespace std;

#include<vector>

#include <bits/stdc++.h>

#include<string>

/\* A binary tree node has data, pointer to left and right children \*/

#include<string>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

node\* newnode(int data)

{

node\* temp = new node;

temp->data = data;

temp->left = NULL;

temp->right = NULL;

return (temp);

}

void pre(struct node\*my){

if(my!=NULL){

cout<<my->data<<" ";

pre(my->left);

pre(my->right);

}

}

int find(string str,int lb,int ub){

if(lb>ub)return -1;

stack<char>s;

int i;

for(i=lb;i<=ub;i++){

if(str[i]=='(')

s.push('(');

else if(str[i]==')'){

if(s.top()=='('){

s.pop();

if(s.empty())

return i;

}

}

}

return -1;

}

struct node\*ans(string str,int l,int u){

if(l>u)

return NULL;

struct node\*my=newnode(str[l]-'0');

int ind=-1;

if(l+1<=u&&str[l+1]=='(')

ind=find(str,l+1,u);

if(ind!=-1){

my->left=ans(str,l+2,ind-1);

my->right=ans(str,ind+2,u-1);

}

return my;

}

int main()

{

string str = "4(2(3)(1))(6(5))";

struct node\*my=ans(str,0,str.length()-1);

pre(my);

return 0;

}