Code :

#include<bits/stdc++.h>

using namespace std;

int n,board[100][100];

const int moves[8][2]={{2,1},{1,2},{-1,2},{-2,1},{-2,-1},{-1,-2},{1,-2},{2,-1}};

bool safe(int x,int y){

return(x>=0 && y>=0 && x<n && y<n && board[x][y]==-1);

}

void printsol(){

cout<<"Solution : "<<endl;

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

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

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

}

cout<<endl;

}

}

bool knightalg(int x, int y, int c){

if(c==n\*n){

return true;

}

for(int i=0; i<8; i++){

int nextx=x+moves[i][0];

int nexty=y+moves[i][1];

if(safe(nextx,nexty)){

board[nextx][nexty]=c;

if(knightalg(nextx,nexty,c+1)){

return true;

}

else{

board[nextx][nexty]=-1;

}

}

}

return false;

}

int main(){

cin>>n;

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

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

board[i][j]=-1;

}

}

int sx=0,sy=0;

board[sx][sy]=0;

if(knightalg(sx,sy,1)){

printsol();

}

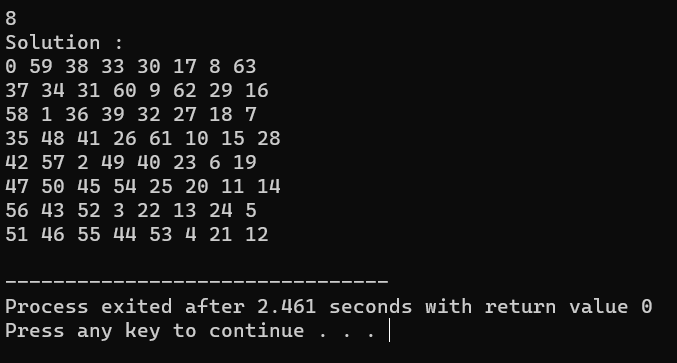
else{

cout<<"No solution!";

}

}

Input/Output :



Code :

#include<bits/stdc++.h>

using namespace std;

int subset\_count=0;

void subset\_sum(int list[], int n, int sum, int s\_index, int t\_sum){

if(t\_sum==sum){

subset\_count++;

if(s\_index<n){

subset\_sum(list,n,sum-list[s\_index-1],s\_index,t\_sum);

}

}

else{

for(int i=s\_index; i<n; i++){

subset\_sum(list,n,sum+list[i],i+1,t\_sum);

}

}

}

int main(){

int n, sum;

cin>>n;

int list[n];

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

cin>>list[i];

}

cin>>sum;

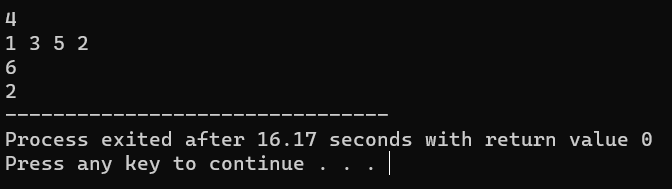
subset\_sum(list,n,0,0,sum);

cout<<subset\_count;

return 0;

}

Input/Output :



Code:

#include<iostream>

using namespace std;

const int s=4;

int graph[s][s],color[s];

bool safe(int n, int p){

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

if(graph[p][i]==1 && n==color[i]){

return false;

}

}

return true;

}

bool malg(int m,int p){

if(p==s){

return true;

}

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

if(safe(i,p)){

color[p]=i;

if(malg(m,p+1)){

return true;

}

color[p]=0;

}

}

return false;

}

int main(){

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

for(int j=0; j<s; j++){

cin>>graph[i][j];

}

}

int m;

cin>>m;

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

color[i]=0;

}

if(malg(m,0)){

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

cout<<color[i]<<" ";

}

}

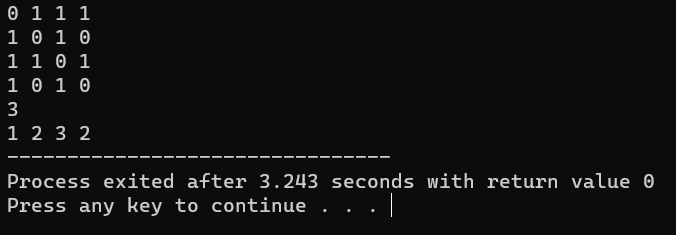
else{

cout<<"No";

}

}

Input/Output :



Code :

#include <iostream>

using namespace std;

int V, graph[100][100],path[100];

bool isSafe(int v, int pos) {

if (graph[path[pos - 1]][v] == 0) {

return false;

}

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

if (path[i] == v) {

return false;

}

}

return true;

}

bool hamiltonianPathUtil( int pos) {

if (pos == V) {

return true;

}

for (int i = 1; i < V; ++i) {

if (isSafe(i, pos)) {

path[pos] = i;

if (hamiltonianPathUtil( pos + 1)) {

return true;

}

path[pos] = -1;

}

}

return false;

}

bool findHamiltonianPath() {

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

path[i] = -1;

}

path[0] = 0;

if (!hamiltonianPathUtil(1)) {

cout << "No Hamiltonian Path exists." << endl;

return false;

}

return true;

}

int main() {

cin >> V;

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

for (int j = 0; j < V; j++) {

cin >> graph[i][j];

}

}

if(findHamiltonianPath()){

cout << "Hamiltonian Path: ";

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

cout << path[i] << " ";

}

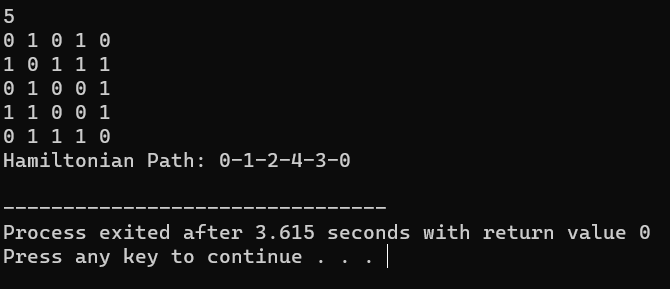
cout << path[0] << endl;

}

return 0;

}

Input/Output :



Code :

#include<iostream>

using namespace std;

const int n=9;

int grid[n][n];

bool safe(int r,int c,int s){

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

if(grid[r][i]==s){

return false;

}

}

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

if(grid[i][c]==s){

return false;

}

}

int sr=r-r%3;

int sc=c-c%3;

for(int i=0; i<3; i++){

for(int j=0; j<3; j++){

if(grid[sr+i][sc+j]==s){

return false;

}

}

}

return true;

}

int printsol(){

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

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

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

}

cout<<endl;

}

}

bool salg(){

int r,c;

bool f=false;

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

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

if(grid[r][c]==0){

f=true;

break;

}

}

if(f){

break;

}

}

if(!f){

return true;

}

for(int i=1; i<=9; i++){

if(safe(r,c,i)){

grid[r][c]=i;

if(salg()){

return true;

}

grid[r][c]=0;

}

}

return false;

}

int main(){

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

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

cin>>grid[i][j];

}

}

if(salg()){

printsol();

}

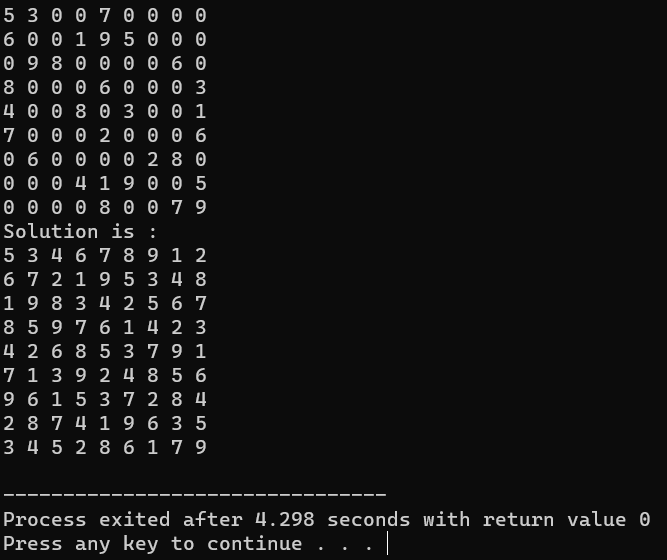
else{

cout<<"No";

}

}

Input/Output :



Code:

#include<iostream>

#include<vector>

#include<cmath>

using namespace std;

vector<int> primeNo;

vector<int> myset;

bool isprime(int x){

int sqroot =sqrt(x);

if(x==1)

return false;

for(int i=2;i<=sqroot;i++)

if(x%i==0)

return false;

return true;

}

void printPrimes(){

int length =myset.size();

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

cout<<myset[i]<<"\t";

cout<<endl;

}

void generate(int total,int n,int s,int index){

if(total==s&&myset.size()==n){

printPrimes();

return;

}

if(total>s||index==primeNo.size())

return;

myset.push\_back(primeNo[index]);

generate(total+primeNo[index],n,s,index+1);

myset.pop\_back();

generate(total,n,s,index+1);

}

void prime(int n,int s,int p){

for(int i=p+1;i<=s;i++){

if(isprime(i))

primeNo.push\_back(i);

}

if(primeNo.size() < n){

return;

}

generate(0,n,s,0);

}

int main(){

int s,n,p;

cin>>s>>n>>p;

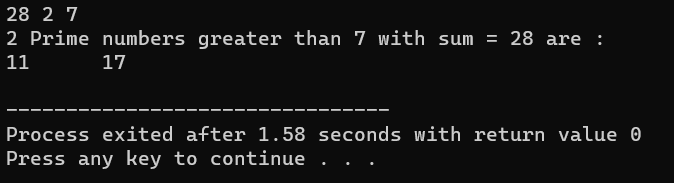
cout<<n<<" Prime numbers greater than "<<p<<" with sum = "<<s<<" are :\n";

prime(n,s,p);

return 0;

}

Input/Output :



Code:

#include<iostream>

using namespace std;

int sundaram(int n){

int N=(n-1)/2;

bool mark[N+1];

for(int i=1; i<=N;i++){

for(int j=1; (i+j+2\*i\*j)<=N; j++){

mark[i+j+2\*i\*j]=true;

}

}

if(n>=2){

cout<<"2 ";

}

for(int i=1; i<=N;i++){

if(!mark[i]){

cout<<2\*i+1<<" ";

}

}

}

int main(){

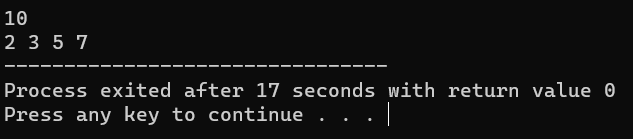
int n;

cin >>n;

sundaram(n);

}

Input/Output :



Code:

#include<iostream>

using namespace std;

int cost[100][100],r,c;

int mincost(){

for(int i=1;i<r; i++){

cost[i][0]+=cost[i-1][0];

}

for(int i=1;i<r; i++){

cost[0][i]+=cost[0][i-1];

}

for(int i=1; i<r; i++){

for(int j=1; j<c; j++){

cost[i][j]+=min(cost[i-1][j],cost[i][j-1]);

}

}

}

int main(){

cin>>r>>c;

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

for(int j=0; j<c; j++){

cin>>cost[i][j];

}

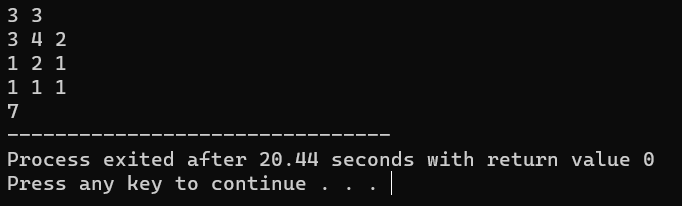
}

mincost();

cout<<cost[r-1][c-1];

}

Input/Output :



Code :

#include <bits/stdc++.h>

using namespace std;

int answer(int arr[], int ranges[][2], int reversals,int index)

{

for (int i = reversals - 1; i >= 0; i--) {

int left = ranges[i][0], right = ranges[i][1];

if (left <= index && right >= index)

index = right + left - index;

}

return arr[index];

}

int main()

{

int n,index;

cin>>n;

int arr[n];

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

cin>>arr[i];

}

int reversals;

cin>>reversals;

int ranges[reversals][2];

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

for(int j=0; j<2; j++){

cin>>ranges[i][j];

}

}

cin>>index;

cout <<"Output : "<< answer(arr, ranges, reversals, index);

return 0;

}

Input/Output :

