# K.madesh

192011133

# 1.Identify the M-th maximum number and Nth minimum number in an array and then find the sum of it and difference of it.

# Test cases: output –

# {16, 16, 16 16, 16}, M = 0, N = 1 (illegal input)

# {0, 0, 0, 0}, M = 1, N = 2 0

# {-12, -78, -35, -42, -85}, M = 3 , N = 3 -7

# {15, 19, 34, 56, 12}, M = 6 , N = -3 (illegal input)

# {85, 45, 65, 75, 95}, M = 5 , N = 2 -20

# Program:

#include<stdio.h>

int main(){

int size=0;

printf("Enter the total no. of elements: ");

scanf("%d",&size);

int arr[size];

printf("Enter the elements: ");

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

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

int n,m;

printf("Enter the m value: ");

scanf("%d",&m);

printf("Enter the n value: ");

scanf("%d",&n);

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

printf("illegal input...!");

else if(m>size || n<0)

printf("illegal input...!");

else{

int temp=0;

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

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

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

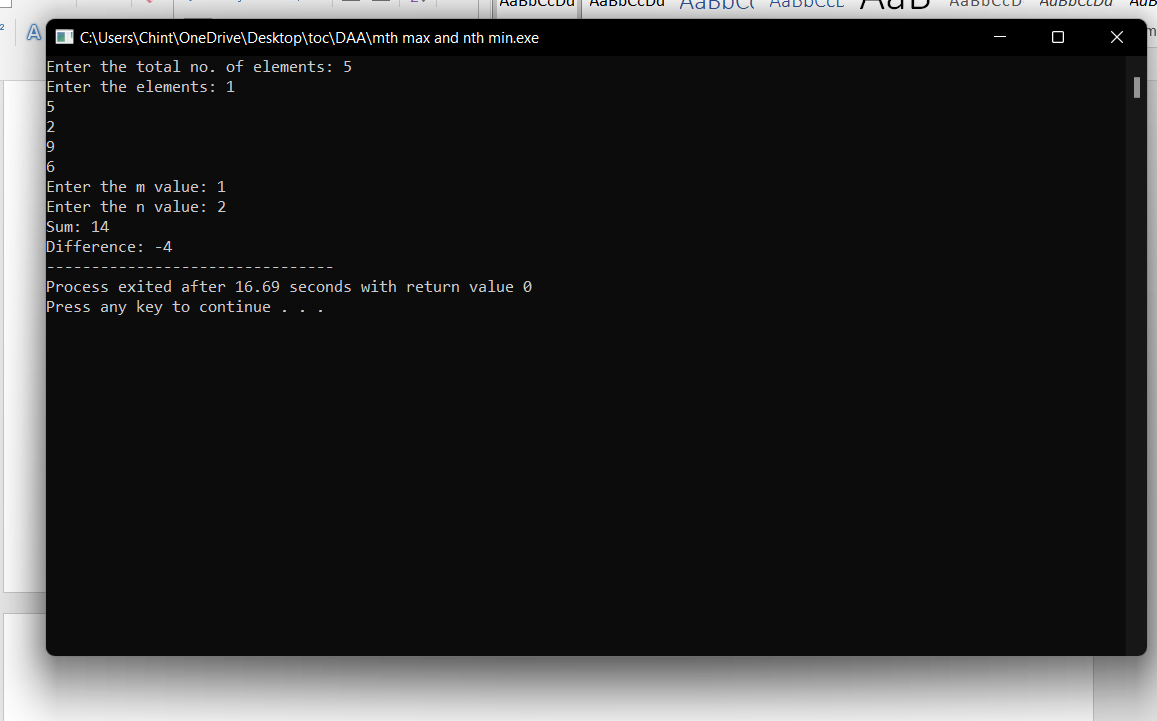
printf("Sum: %d\n",arr[size-m]+arr[n-1]);

printf("Difference: %d",arr[size-m]-arr[n-1]);

}

}

# Output:



1. **Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.integer target. Write a program to search a number in a list using binary search and estimate time complexity**

# Program:

#include<stdio.h>

int main()

{

int c=0;

int n,k,i,low,high,mid,a[50],temp;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter elements:\n");

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

{

c++;

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

}

c++;

printf("Enter Element to search:");

scanf("%d",&k);

low=0; c++;

high=n-1; c++;

mid=low+high/2; c++;

c++;

while(low<=high)

{

c++;

c++;

if(a[mid]<k)

{

low=mid+1; c++;

}

else if(a[mid]==k)

{

printf("\nElement is found at index %d\n",mid);

break;

}

else

{

high=mid-1; c++;

}

mid=(low+high)/2; c++;

}

c++;

c++;

if(low>high)

{

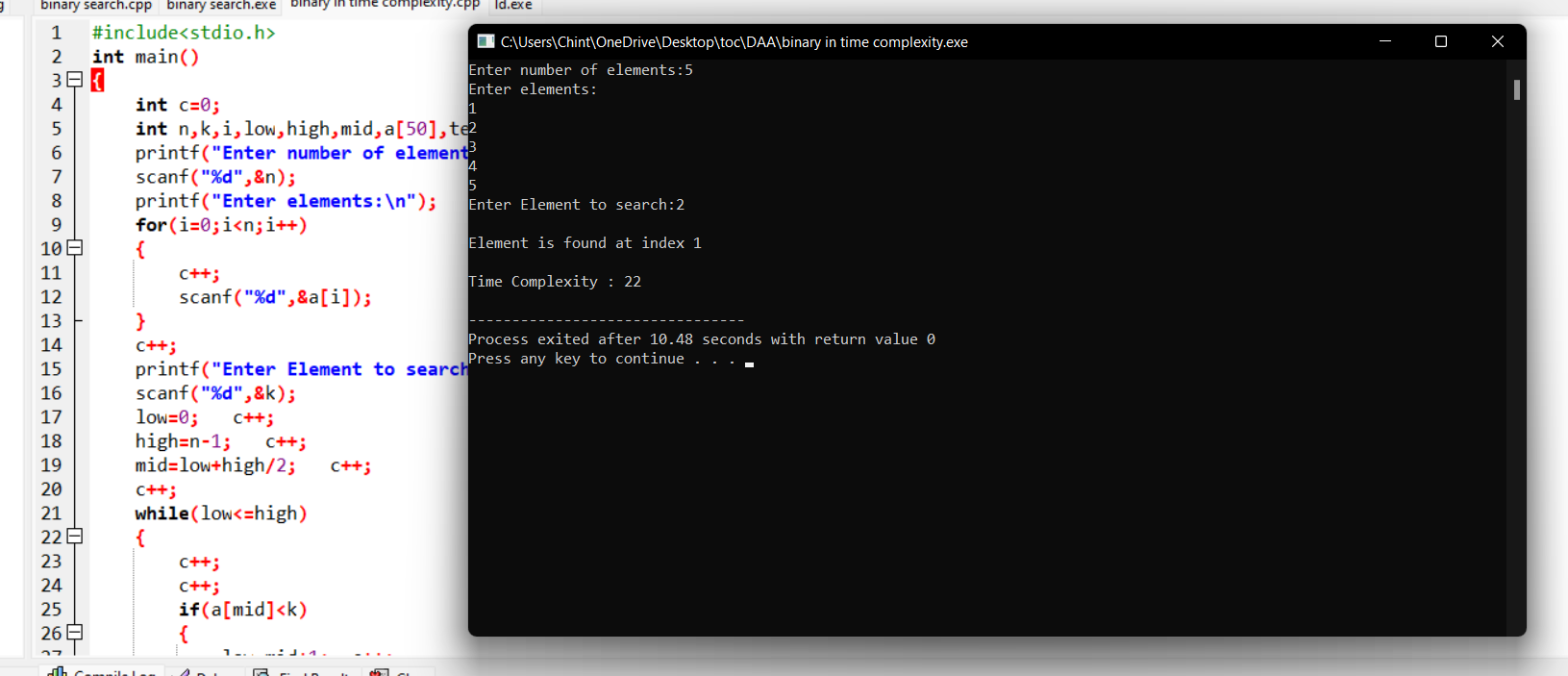
printf("Element is not found\n");

}

printf("\nTime Complexity : %d\n",c);

}

# Output:



1. **Write a program to find the reverse of a given number. Estimate the time complexity for the following inputs**
   1. **1234 b. 6789456 c. 45a34 d – 5926**

# Program:

#include <stdio.h>

int main() {

int n, reverse = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while (n != 0) {

remainder = n % 10;

reverse = reverse \* 10 + remainder;

n /= 10;

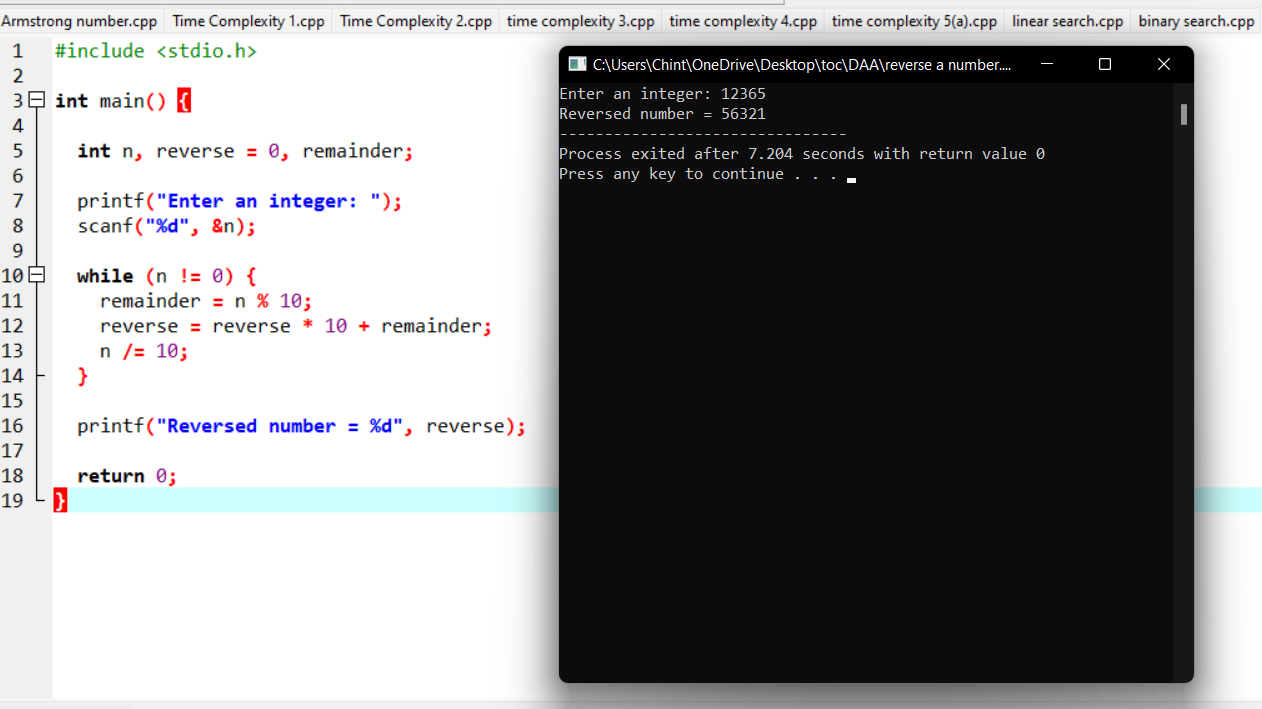
}

printf("Reversed number = %d", reverse);

return 0;

}

# Output:



1. **Write a program to perform sum of subsets problem using backtracking and estimate time complexity. Identify the test cases.**

**A. Set (s) = (6, 2,8,1,5) sum is 9 B.. Set (s) = (6, -4, 7,-1,5, 2,8,1,) sum is 10**

# Program:

#include <stdio.h>

#define TRUE 1

#define FALSE 0

int inc[50],w[50],sum,n,count=0;

void sumset(int i,int wt,int total);

int promising(int i,int wt,int total) {

return(((wt+total)>=sum)&&((wt==sum)||(wt+w[i+1]<=sum)));

}

int main() {

int i,j,n,temp,total=0;

printf("\n Enter how many numbers:\n");

scanf("%d",&n);

printf("\n Enter %d numbers to th set:\n",n);

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

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

total+=w[i];

}

count++;

printf("\n Input the sum value to create sub set:\n");

scanf("%d",&sum);

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

for (j=0;j<n-1;j++)

if(w[j]>w[j+1]) {

temp=w[j];

w[j]=w[j+1];

w[j+1]=temp;

}

count++;

printf("\n The given %d numbers in ascending order:\n",n);

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

printf("%d \t",w[i]);

count++;

if((total<sum))

printf("\n Subset construction is not possible"); else {

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

inc[i]=0;

printf("\n The solution using backtracking is:\n");

sumset(-1,0,total);

}

count++;

}

void sumset(int i,int wt,int total) {

int j;

if(promising(i,wt,total)) {

if(wt==sum) {

printf("\n{\t");

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

if(inc[j])

printf("%d\t",w[j]);

printf("}\n");

} else {

inc[i+1]=TRUE;

sumset(i+1,wt+w[i+1],total-w[i+1]);

inc[i+1]=FALSE;

sumset(i+1,wt,total-w[i+1]);

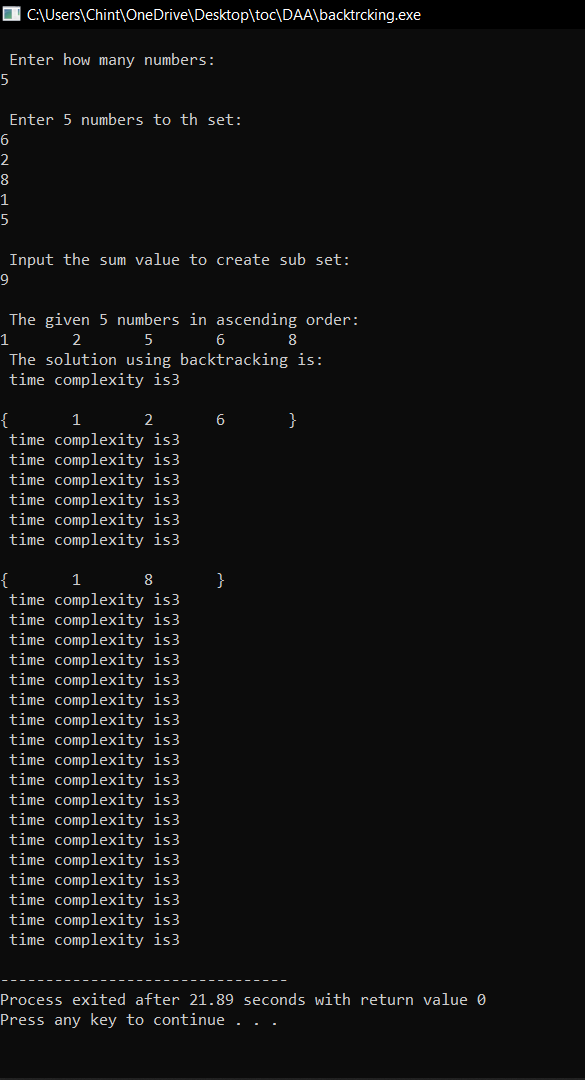
}

}

printf(" time complexity is%d\n",count);

}

# Output:



**5. Write a program to check the given is Armstrong or not.**

**The k-digit number N is an Armstrong number if and only if the k-th power of**

**each digit sums to N.**

**Given a positive integer N, return true if and only if it is an Armstrong number.**

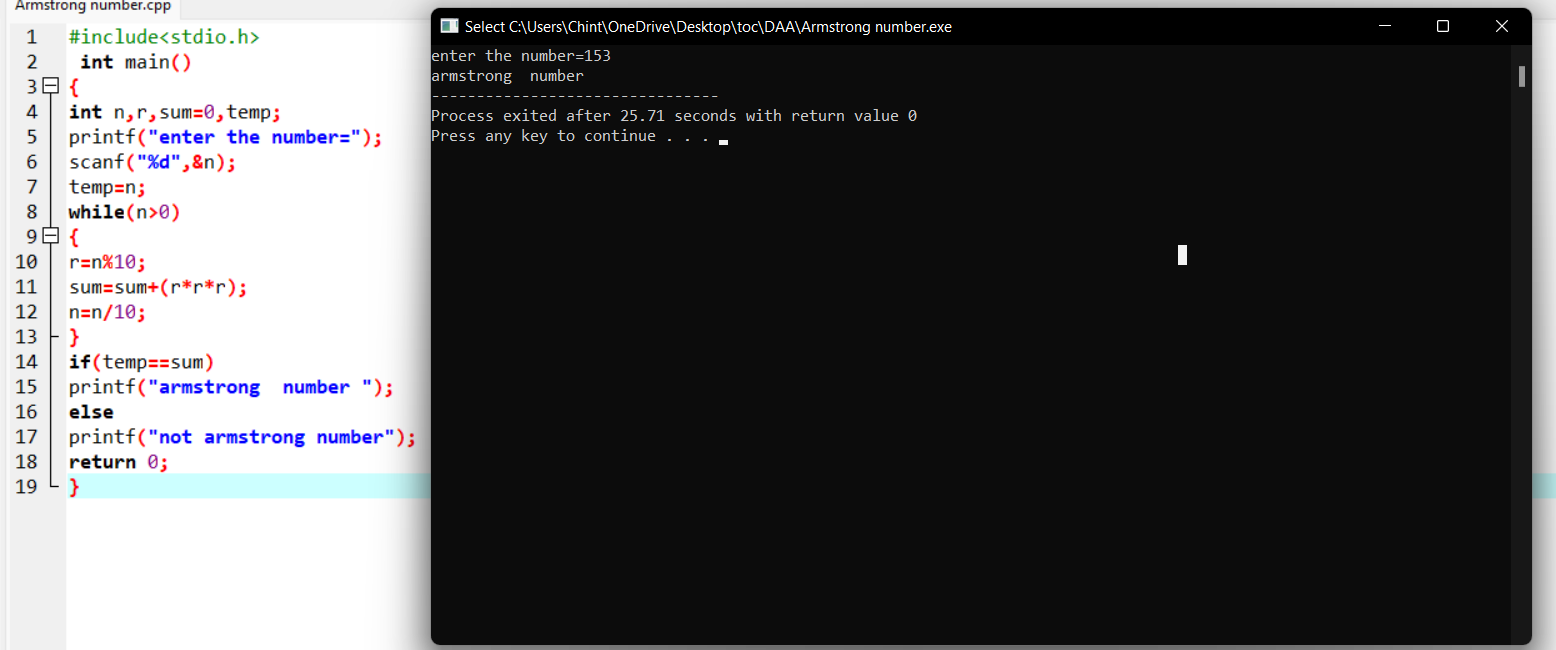
**Input : 153 Input : 419**

**Output : True Output : False**

# Program:

1. #include<stdio.h>
2. **int** main()
3. {
4. **int** n,r,sum=0,temp;
5. printf("enter the number=");
6. scanf("%d",&n);
7. temp=n;
8. **while**(n>0)
9. {
10. r=n%10;
11. sum=sum+(r\*r\*r);
12. n=n/10;
13. }
14. **if**(temp==sum)
15. printf("armstrong  number ");
16. **else**
17. printf("not armstrong number");
18. **return** 0;
19. }

# Output:



6. **Write a C program to perform Strassen’s Matrix Multiplication for the 2\*2 matrix elements.**

**And Estimate time complexity.**

**A Matrix= (3, 5,-4, 7) B Matrix – (9,-2, 4, 8)**

# Program:

#include<stdio.h>

int main(){

int a[2][2], b[2][2], c[2][2], i, j,count=0;

int m1, m2, m3, m4 , m5, m6, m7;

printf("Enter the 4 elements of first matrix:");

count++;

for(i = 0;i < 2; i++)

{

count++;

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

{

count++;

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

}

}

count++;

count++;

printf("Enter the 4 elements of second matrix: ");

for(i = 0; i < 2; i++)

{

count++;

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

{

count++;

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

}

}

count++;

count++;

printf("\nThe first matrix is\n");

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

count++;

printf("\n");

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

count++;

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

}

}

count++;

count++;

printf("\nThe second matrix is\n");

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

count++;

printf("\n");

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

count++;

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

}

}

count++;

count++;

m1= (a[0][0] + a[1][1]) \* (b[0][0] + b[1][1]);

count++;

m2= (a[1][0] + a[1][1]) \* b[0][0];

count++;

m3= a[0][0] \* (b[0][1] - b[1][1]);

count++;

m4= a[1][1] \* (b[1][0] - b[0][0]);

count++;

m5= (a[0][0] + a[0][1]) \* b[1][1];

count++;

m6= (a[1][0] - a[0][0]) \* (b[0][0]+b[0][1]);

count++;

m7= (a[0][1] - a[1][1]) \* (b[1][0]+b[1][1]);

count++;

c[0][0] = m1 + m4- m5 + m7;

count++;

c[0][1] = m3 + m5;

count++;

c[1][0] = m2 + m4;

count++;

c[1][1] = m1 - m2 + m3 + m6;

count++;

printf("\nAfter multiplication using Strassen's algorithm \n");

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

count++;

printf("\n");

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

count++;

printf("%d\t", c[i][j]);

}

}

count++;

count++;

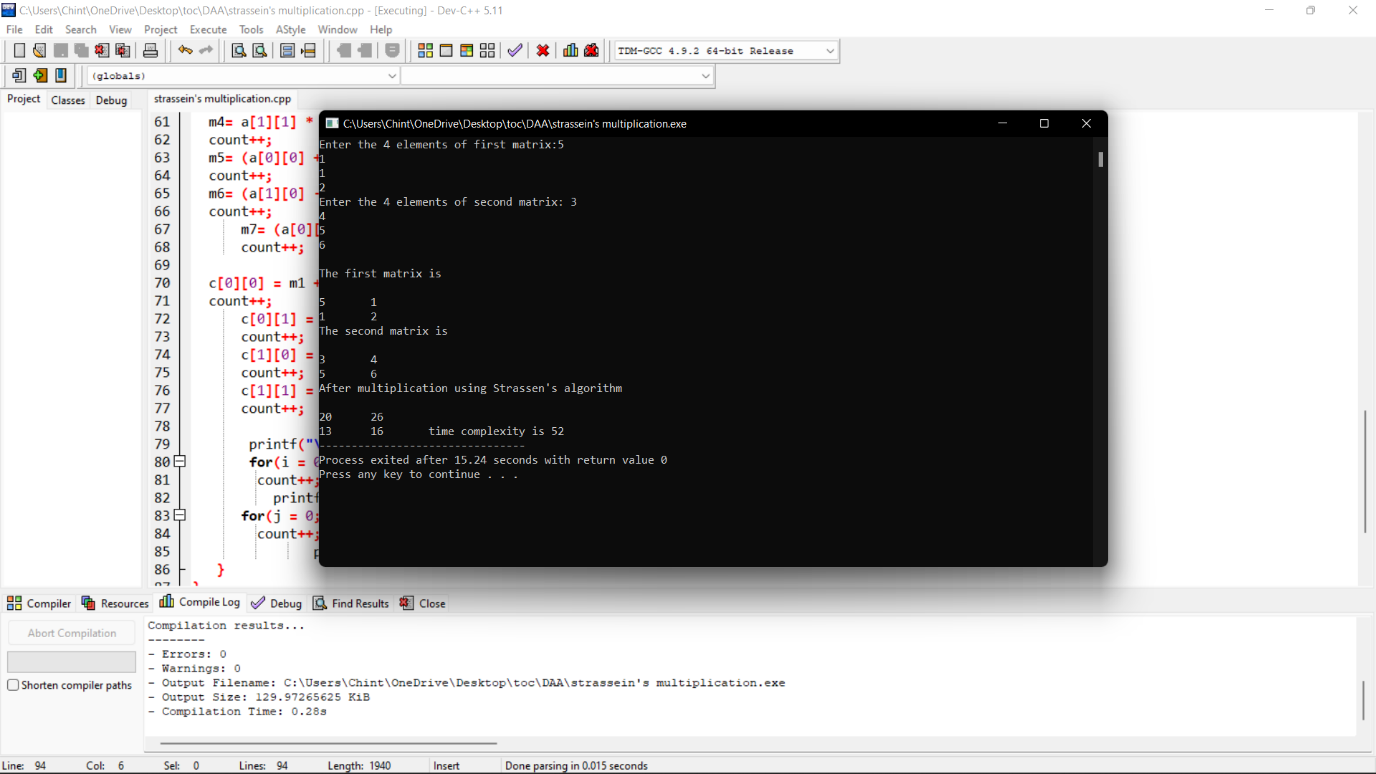
printf(" time complexity is %d",count);

return 0;

}

}

# Output:



3**. Write a program to generate all the reverse of a prime should be prime**

**( example 907 is prime and reverse 709 is also prime )**

**Generate all the no’s upto N and estimate time complexity.**

# Program:

#include <stdio.h>

int main()

{

int count=0;

int n, reverse,sum=0 , flag;

printf("Enter the prime number:");

scanf("%d",&n);

while(n!=0)

{

count++;

reverse = n%10;

count++;

sum = sum\*10 + reverse;

count++;

n= n/10;

count++;

}

printf("\n");

flag = 0;

for (int j = 2; j <= sum / 2; j++)

{

count++;

if ((sum % j) == 0)

{

count++;

flag = 1;

break;

}

}

if (flag == 0)

{

count++;

printf("%d is also prinme number",sum);

}

else

{

count++;

printf("%d is Not Prime number\n",sum);

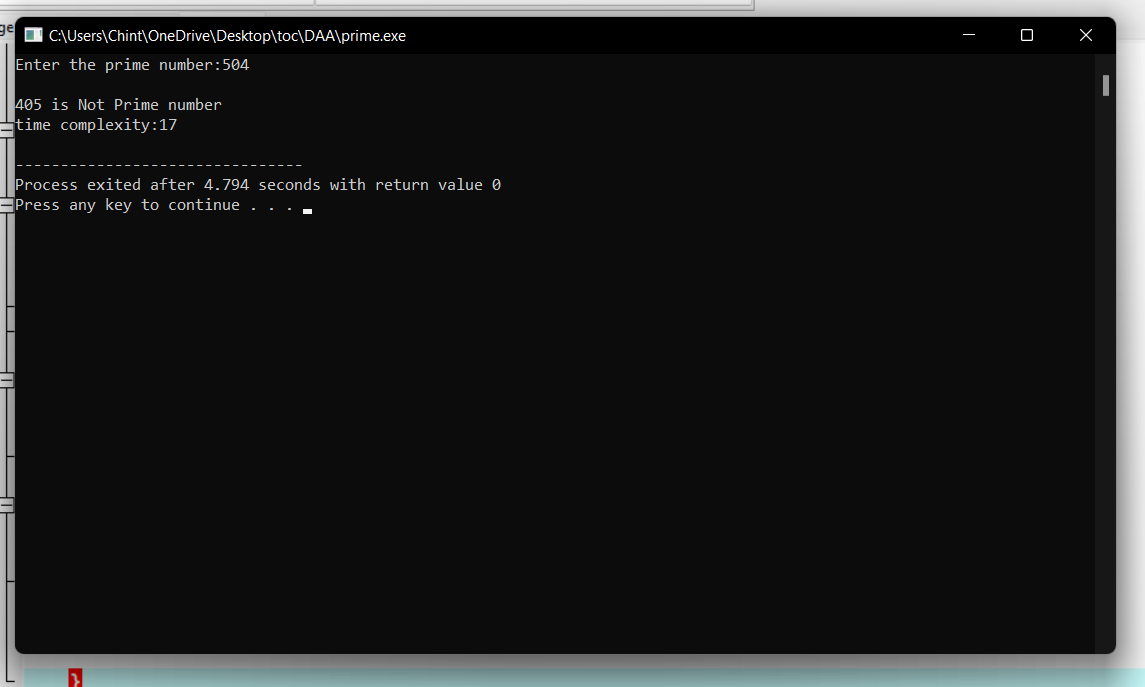
}

count++;

printf("time complexity:%d\n",count);

}

# Output:



8. **Let there be N workers and N jobs. Any worker can be assigned to perform any job, incurring some**

**cost that may vary depending on the work-job assignment. It is required to perform all jobs by**

**assigning exactly one worker to each job and exactly one job to each agent in such a way that the**

**total cost of the assignment is minimized. Write a program to solve a assignment problem for the**

**given data sets using branch and bound.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Job 1** | **Job 2** | **Job 3** | **Job 4** |
| **Person A** | **12** | **8** | **9** | **10** |
| **Person B** | **11** | **10** | **10** | **9** |
| **Person C** | **9** | **11** | **8** | **12** |
| **Person D** | **11** | **9** | **23** | **7** |

**9. Compute the program to find the GCD of two numbers. And also find the finf of time Recursion**

**used to estimate time complexity.**

**Perform the test cases for the given set of no’s**

1. **(36,48) B. (144, 90) C. (-56,88) D. (84,84)**

# Program:

#include <stdio.h>

int main()

{

int n1, n2, i, GCD\_Num;

int count=0;

printf ( " Enter any two numbers: \n ");

scanf ( "%d %d", &n1, &n2);

for( i = 1; i <= n1 && i <= n2; ++i)

{

count++;

if (n1 % i ==0 && n2 % i == 0)

GCD\_Num = i;

count++;

}

count++;

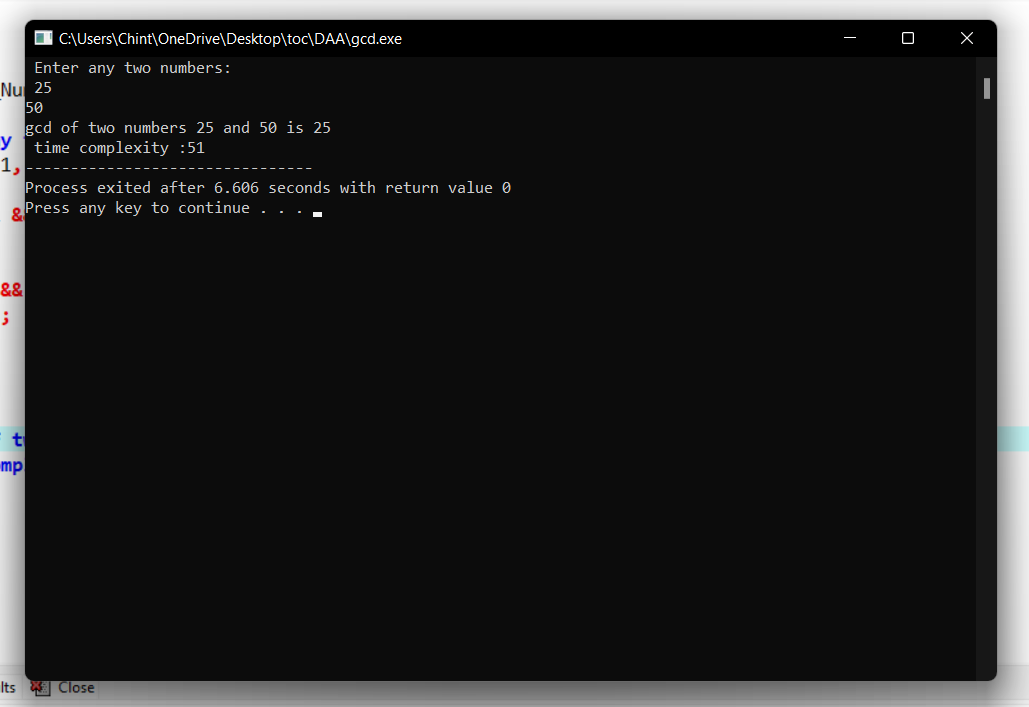
printf ("gcd of two numbers %d and %d is %d \n ", n1, n2, GCD\_Num);

printf("time complexity :%d ",count);

return 0;

}

# Output:



**10. Using Divide and Conquer strategy to find Max and Min value in the list and estimate time**

**complexity.**

**Testing Condition – Count the number of times in Comparion to find Min\_Max value**

**in a list n for the given set of elements.**

1. **(23,45,6,8,-9,44,7,8)**
2. **(8,-5,7,2,6,0,1,9)**
3. **(45, y, 9, 8,4, 7,11, 22,16)**

# Program:

#include<stdio.h>

#include<stdio.h>

int max, min;

int a[100];

void maxmin(int i, int j)

{

int max1, min1, mid;

if(i==j)

{

max = min = a[i];

}

else

{

if(i == j-1)

{

if(a[i] <a[j])

{

max = a[j];

min = a[i];

}

else

{

max = a[i];

min = a[j];

}

}

else

{

mid = (i+j)/2;

maxmin(i, mid);

max1 = max; min1 = min;

maxmin(mid+1, j);

if(max <max1)

max = max1;

if(min > min1)

min = min1;

}

}

}

int main ()

{

int i, num;

printf ("\nEnter the total number of numbers : ");

scanf ("%d",&num);

printf ("Enter the numbers : \n");

for (i=1;i<=num;i++)

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

max = a[0];

min = a[0];

maxmin(1, num);

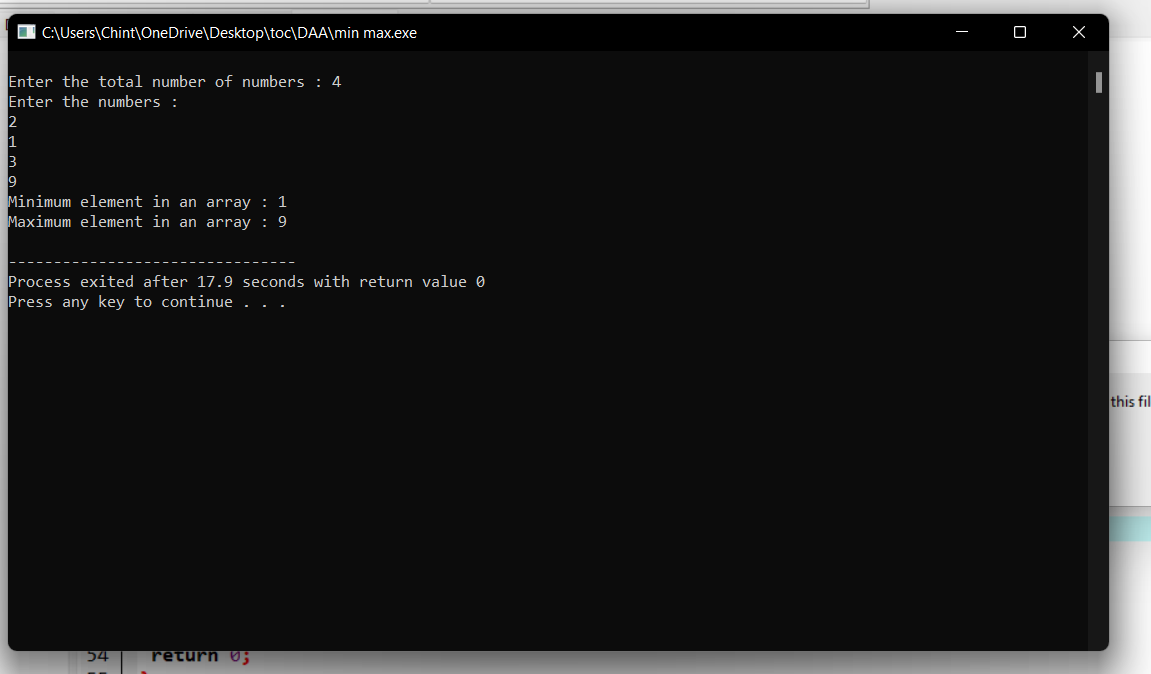
printf ("Minimum element in an array : %d\n", min);

printf ("Maximum element in an array : %d\n", max);

return 0;

}

# Output:



**11. Generate a program for Pascal triangle.**

**Estimate the time complexity for the row=5**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | **1** |  |  |  |  |
|  |  |  | **1** |  | **1** |  |  |  |
|  |  | **1** |  | **2** |  | **1** |  |  |
|  | **1** |  | **3** |  | **3** |  | **1** |  |
| **1** |  | **4** |  | **6** |  | **4** |  | **1** |
|  |  |  |  |  |  |  |  |  |

# Program:

#include<stdio.h>

int main()

{

int rows, coef = 1, space, i, j;

int count=0;

printf("Enter the number of rows: ");

scanf("%d", &rows);

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

{

count++;

for (space = 1; space <= rows - i; space++)

printf(" ");

count++;

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

{

count++;

if(j == 0 || i == 0){

coef = 1;

count++;

}

else

{

coef = coef \* (i - j + 1) / j;

}

count++;

printf("%4d", coef);

}

printf("\n");

count++;

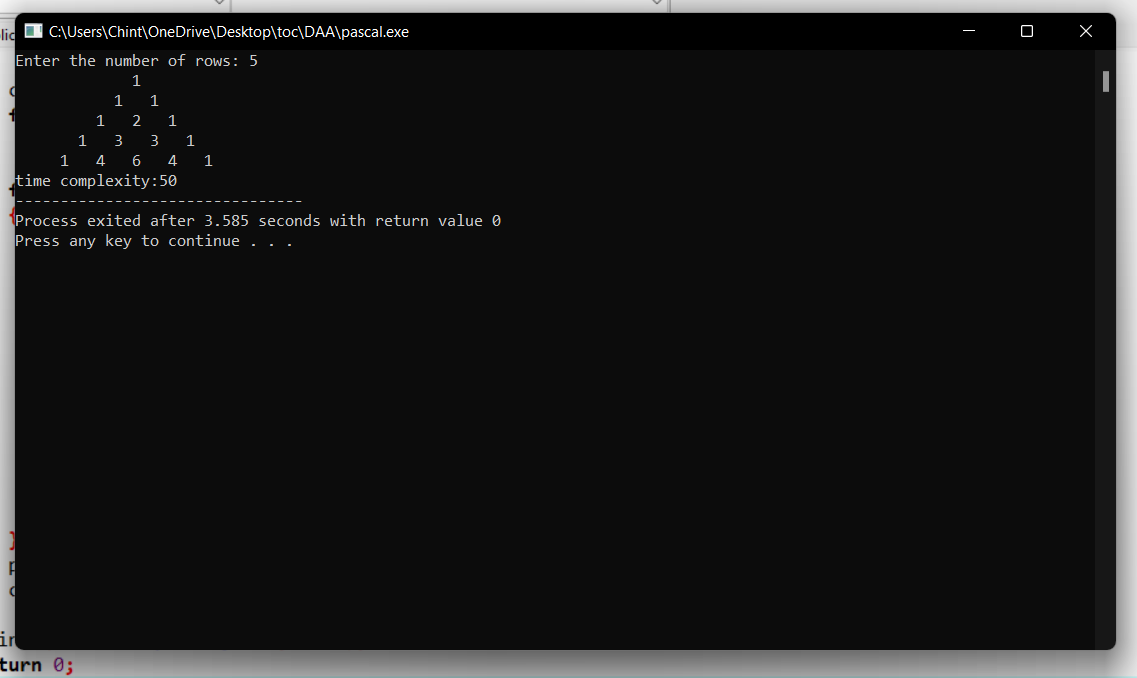
}

printf("time complexity:%d",count);

return 0;

}

# Output:



**12. Compute Binomial coefficient for n=8, k=8 using dynamic programming**

**Using condition such as**

**I nCk =1 if k=0 or n=k**

**II nCk – (n-1)Ck-1 + (n-1)Ck for n>k>0**

# Program:

#include <stdio.h>

int count=0;

int bin\_table(int val) {

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

count++;

printf("%2d", i);

int num = 1;

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

count++;

if (i != 0 && j != 0)

num = num \* (i - j + 1) / j;

count++;

printf("%4d", num);

}

printf("\n");

count++;

}

}

int main() {

int value = 5;

bin\_table(value);

printf("Time complexity:%d",count);

return 0;

}

# Output:

0 1

1 1 1

2 1 2 1

3 1 3 3 1

4 1 4 6 4 1

5 1 5 10 10 5 1

Time complexity:54

--------------------------------

**13. Write a program to find the largest element value in an array. Estimate the time complexity and no of**

**comparison for the given set of values.**

**Program:**

#include <stdio.h>

int main() {

int n;

int count=0;

double arr[100];

printf("Enter the number of elements (1 to 100): ");

scanf("%d", &n);

count++;

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

count++;

printf("Enter number%d: ", i + 1);

scanf("%lf", &arr[i]);

}

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

count++;

if (arr[0] < arr[i]) {

arr[0] = arr[i];

}

count++;

}

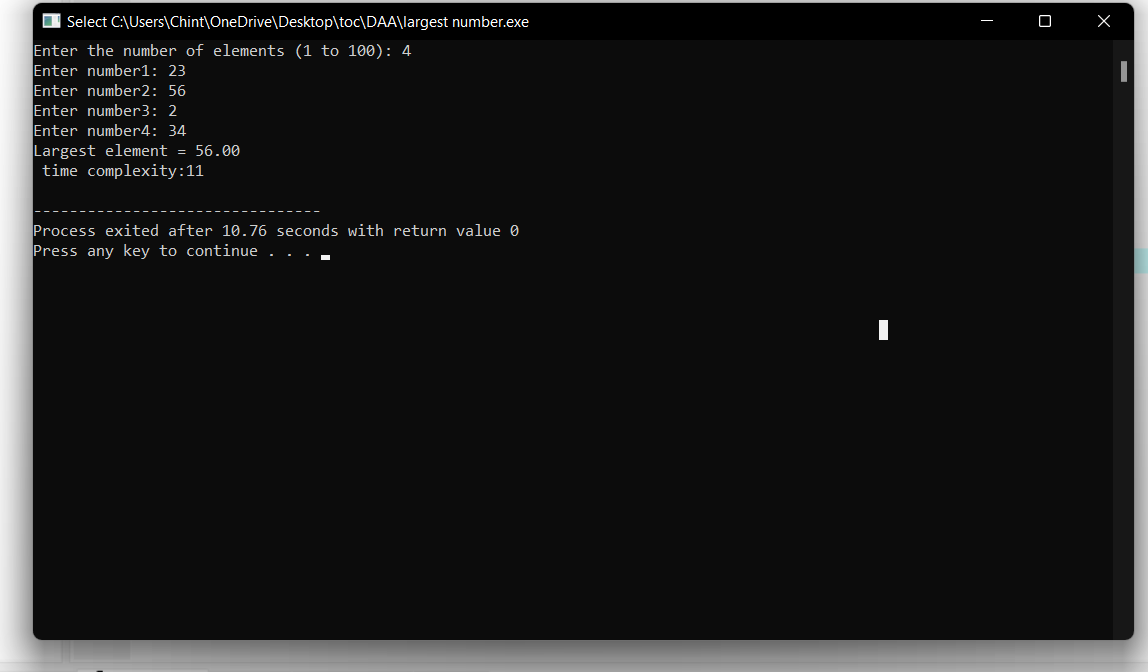
printf("Largest element = %.2lf \n ", arr[0]);

printf("time complexity:%d\n",count);

return 0;

}

# Output:



14**. Consider a two integer arrays nums1 and nums2, sorted in non-increasing order and two integers m**

**and n, representing the number of elements in nums1 and nums2 respectively. Write a program to**

**Merge them into a single array using Merge Sort. Derive time complexity of merge sort.**

**Program:**

#include <stdio.h>

#include <stdlib.h>

// Merges two subarrays of arr[].

// First subarray is arr[l..m]

// Second subarray is arr[m+1..r]

void merge(int arr[], int l,

int m, int r)

{

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

// Create temp arrays

int L[n1], R[n2];

// Copy data to temp arrays

// L[] and R[]

for (i = 0; i < n1; i++)

L[i] = arr[l + i];

for (j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

// Merge the temp arrays back

// into arr[l..r]

// Initial index of first subarray

i = 0;

// Initial index of second subarray

j = 0;

// Initial index of merged subarray

k = l;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

// Copy the remaining elements

// of L[], if there are any

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

// Copy the remaining elements of

// R[], if there are any

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

// l is for left index and r is

// right index of the sub-array

// of arr to be sorted

void mergeSort(int arr[],

int l, int r)

{

if (l < r)

{

// Same as (l+r)/2, but avoids

// overflow for large l and h

int m = l + (r - l) / 2;

// Sort first and second halves

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

// UTILITY FUNCTIONS

// Function to print an array

void printArray(int A[], int size)

{

int i;

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

printf("%d ", A[i]);

printf("\n");

}

// Driver code

int main()

{

int arr[] = {12, 11, 13, 5, 6, 7};

int arr\_size = sizeof(arr) / sizeof(arr[0]);

printf("Given array is \n");

printArray(arr, arr\_size);

mergeSort(arr, 0, arr\_size - 1);

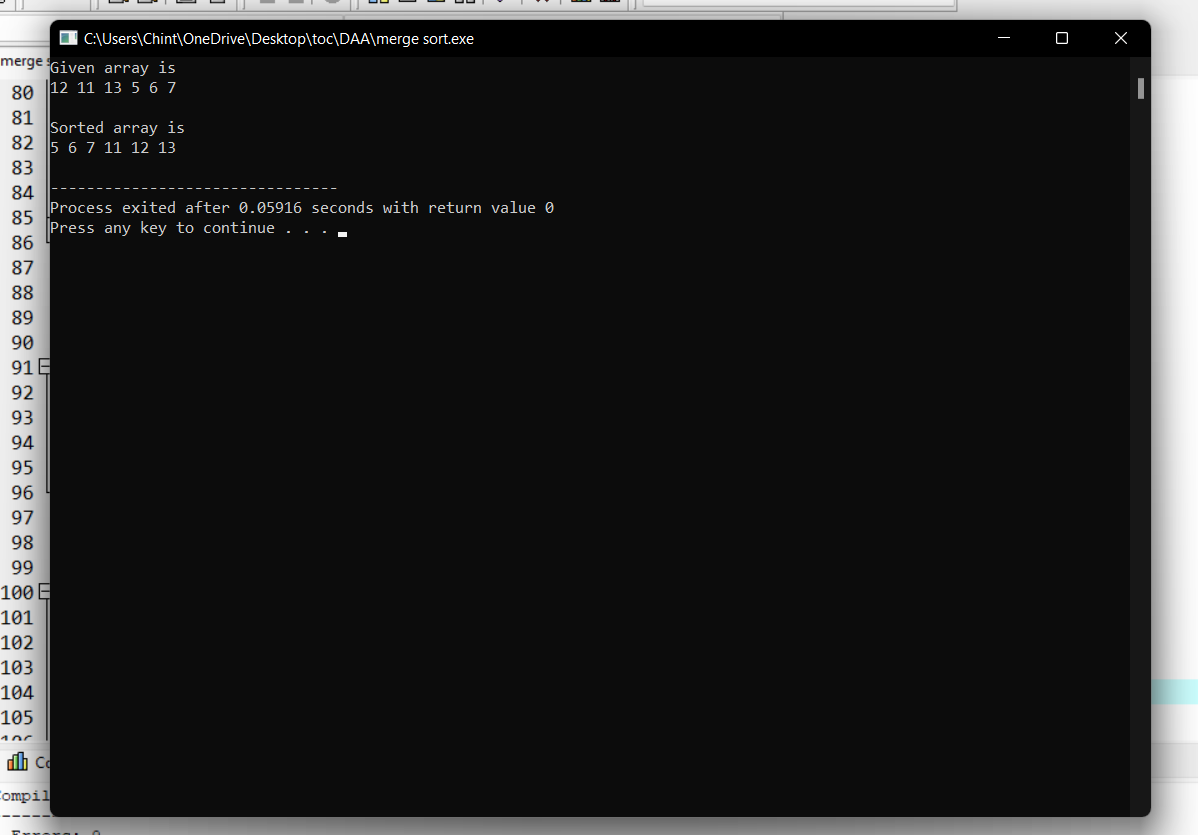
printf("\nSorted array is \n");

printArray(arr, arr\_size);

return 0;

}

**Output:**



1**5.Write a program to find all pair shortest path using Floyds technique and to estimate time complexity.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **A** | **0** | **8** | **7** | **8** |
| **B** | **9** | **0** | **11** | **12** |
| **C** | **10** | **9** | **0** | **11** |
| **D** | **8** | **10** | **11** | **0** |

# Program:

#include <stdio.h>

// defining the number of vertices

#define nV 4

#define INF 999

void printMatrix(int matrix[][nV]);

// Implementing floyd warshall algorithm

void floydWarshall(int graph[][nV]) {

int matrix[nV][nV], i, j, k;

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

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

matrix[i][j] = graph[i][j];

// Adding vertices individually

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

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

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

if (matrix[i][k] + matrix[k][j] < matrix[i][j])

matrix[i][j] = matrix[i][k] + matrix[k][j];

}

}

}

printMatrix(matrix);

}

void printMatrix(int matrix[][nV]) {

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

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

if (matrix[i][j] == INF)

printf("%4s", "INF");

else

printf("%4d", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int graph[nV][nV] = {{0, 8, INF, 8},

{9, 0, INF, 12},

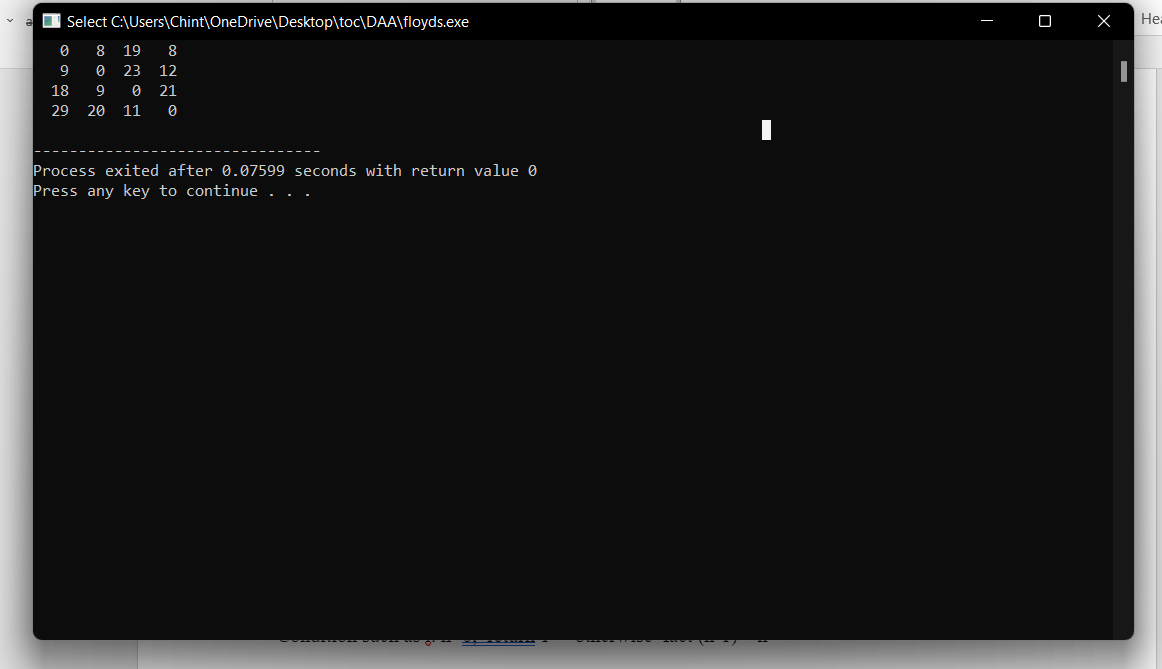
{INF, 9, 0, INF},

{INF, INF, 11, 0}};

floydWarshall(graph);

}

# Output:



1. N = 0.2

**.16.Write a program for to perform liner search and estimate time complexity. Compute the**

**amount of time for completion.**

**Input series**

**A = (56,89,7,13,75, 23, 8, 12) Key element 75**

**B= (89,45 -23,45,0, 44, 2) Key element 0**

**C= (45,67,56,A,34,-2,100) Key element 90**

# Program:

#include<stdio.h>

int main()

{

int c=0;

int n,k,i,j,f=0,a[50];

c++;

printf("Enter number of elements:");

scanf("%d",&n);

printf("Enter elements:\n");

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

{

c++;

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

}

c++;

printf("Enter Element to search:");

scanf("%d",&k);

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

{

c++;

c++;

if(k==a[i])

{

printf("Element is found at index %d\n",i);

f=1;

c++;

}

}

c++;

c++;

if(f==0)

{

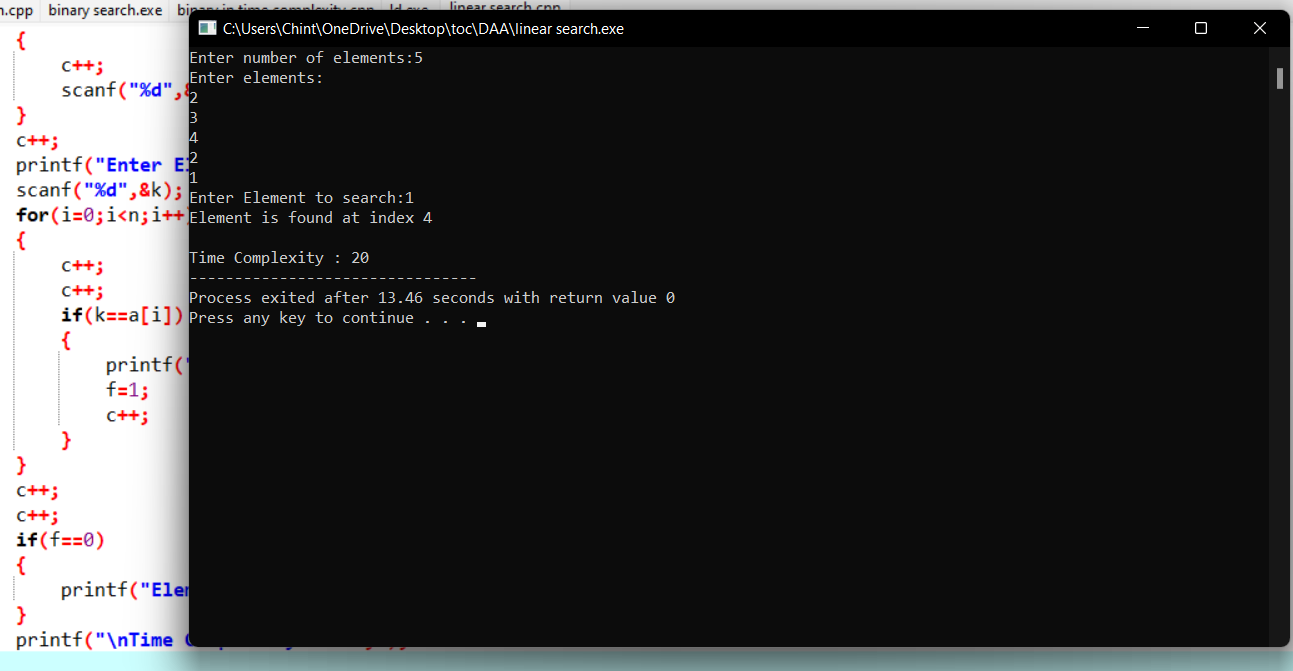
printf("Element is not found");

}

printf("\nTime Complexity : %d",c);

}

# Output:



**17. Write a program to find the factorial (fact)of a number and to estimate time complexity.**

**Condition such as i. n=0, return 1 otherwise fact (n-1) \* n**

**Program:**

#include <stdio.h>

int main() {

int n, i;

int count=0;

unsigned long long fact = 1;

printf("Enter an integer: ");

scanf("%d", &n);

count++;

if (n < 0)

printf("Error! Factorial of a negative number doesn't exist.");

else {

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

fact \*= i;

count++;

}

printf("Factorial of %d = %llu \n ", n, fact);

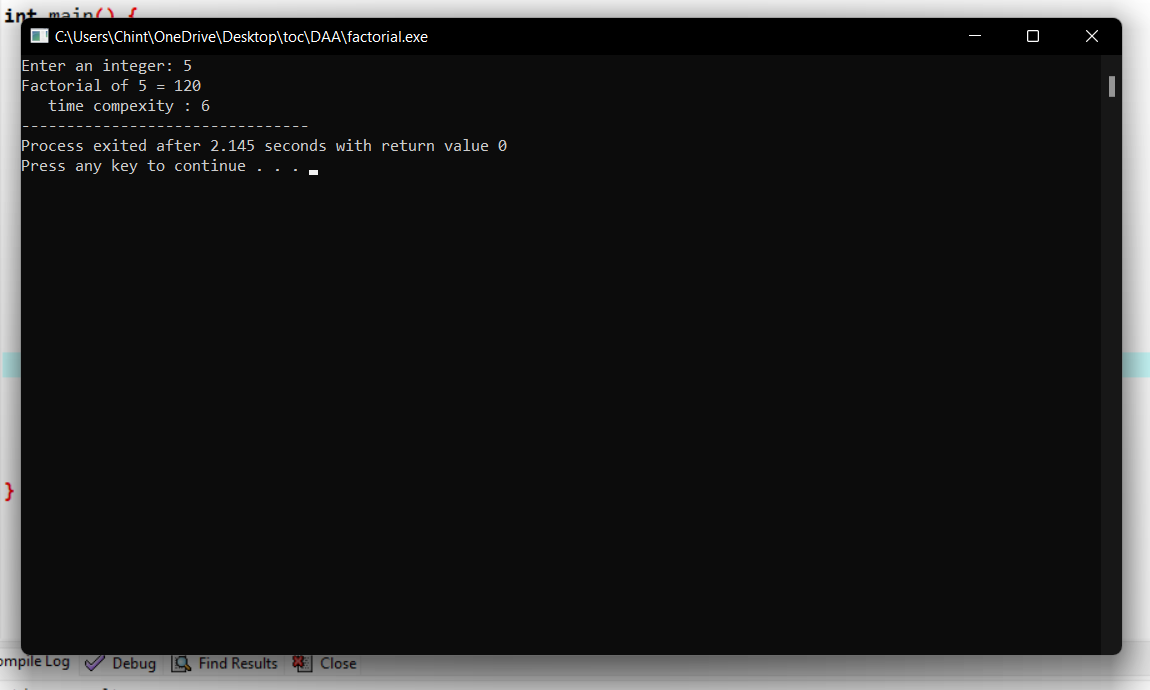
printf(" time compexity : %d ",count);

}

return 0;

**}**

**Output:**



**18.Write a program to perform Knapsack problem for the following set of object values.,**

**Knapsack weight 100**

|  |  |  |
| --- | --- | --- |
| **item** | **Weight** | **Profit** |
| **1** | **40** | **80** |
| **2** | **30** | **70** |
| **3** | **20** | **50** |
| **4** | **30** | **80** |
|  |  |  |
|  |  |  |

**Program:**

# include<stdio.h>

void knapsack(int n, float weight[], float profit[], float capacity) {

float x[20], tp = 0;

int i, j, u;

u = capacity;

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

x[i] = 0.0;

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

if (weight[i] > u)

break;

else {

x[i] = 1.0;

tp = tp + profit[i];

u = u - weight[i];

}

}

if (i < n)

x[i] = u / weight[i];

tp = tp + (x[i] \* profit[i]);

printf("\nThe result vector is:- ");

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

printf("%f\t", x[i]);

printf("\nMaximum profit is:- %f", tp);

}

int main() {

float weight[20], profit[20], capacity;

int num, i, j;

float ratio[20], temp;

printf("\nEnter the no. of objects:- ");

scanf("%d", &num);

printf("\nEnter the wts and profits of each object:- ");

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

scanf("%f %f", &weight[i], &profit[i]);

}

printf("\nEnter the capacityacity of knapsack:- ");

scanf("%f", &capacity);

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

ratio[i] = profit[i] / weight[i];

}

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

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

if (ratio[i] < ratio[j]) {

temp = ratio[j];

ratio[j] = ratio[i];

ratio[i] = temp;

temp = weight[j];

weight[j] = weight[i];

weight[i] = temp;

temp = profit[j];

profit[j] = profit[i];

profit[i] = temp;

}

}

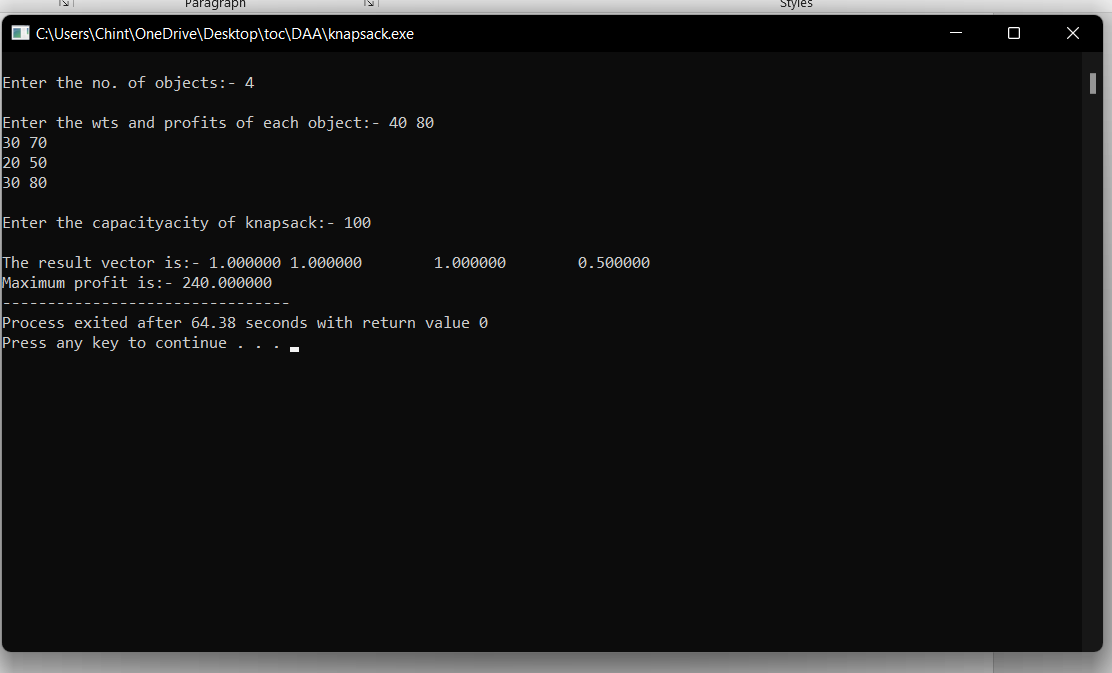
}

knapsack(num, weight, profit, capacity);

return(0);

}

**Output:**



**19.Write a program to print the first n perfect numbers. (Hint Perfect number means a positive integer that is equal to the sum of its proper divisors)**

**Sample Input:**

**N = 3**

**Sample Output:**

**First 3 perfect numbers are: 6 , 28 , 496**

**Test Cases:**

1. **N = 0**
2. **N = 5**
3. **N = -2**
4. **N = -5**

**Program:**

#include <stdio.h>

int main()

{

int i, j, end, sum;

/\* Input upper limit to print perfect number \*/

printf("Enter upper limit: ");

scanf("%d", &end);

printf("All Perfect numbers between 1 to %d:\n", end);

/\* Iterate from 1 to end \*/

for(i=1; i<=end; i++)

{

sum = 0;

/\* Check whether the current number i is Perfect number or not \*/

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

{

if(i % j == 0)

{

sum += j;

}

}

/\* If the current number i is Perfect number \*/

if(sum == i)

{

printf("%d, ", i);

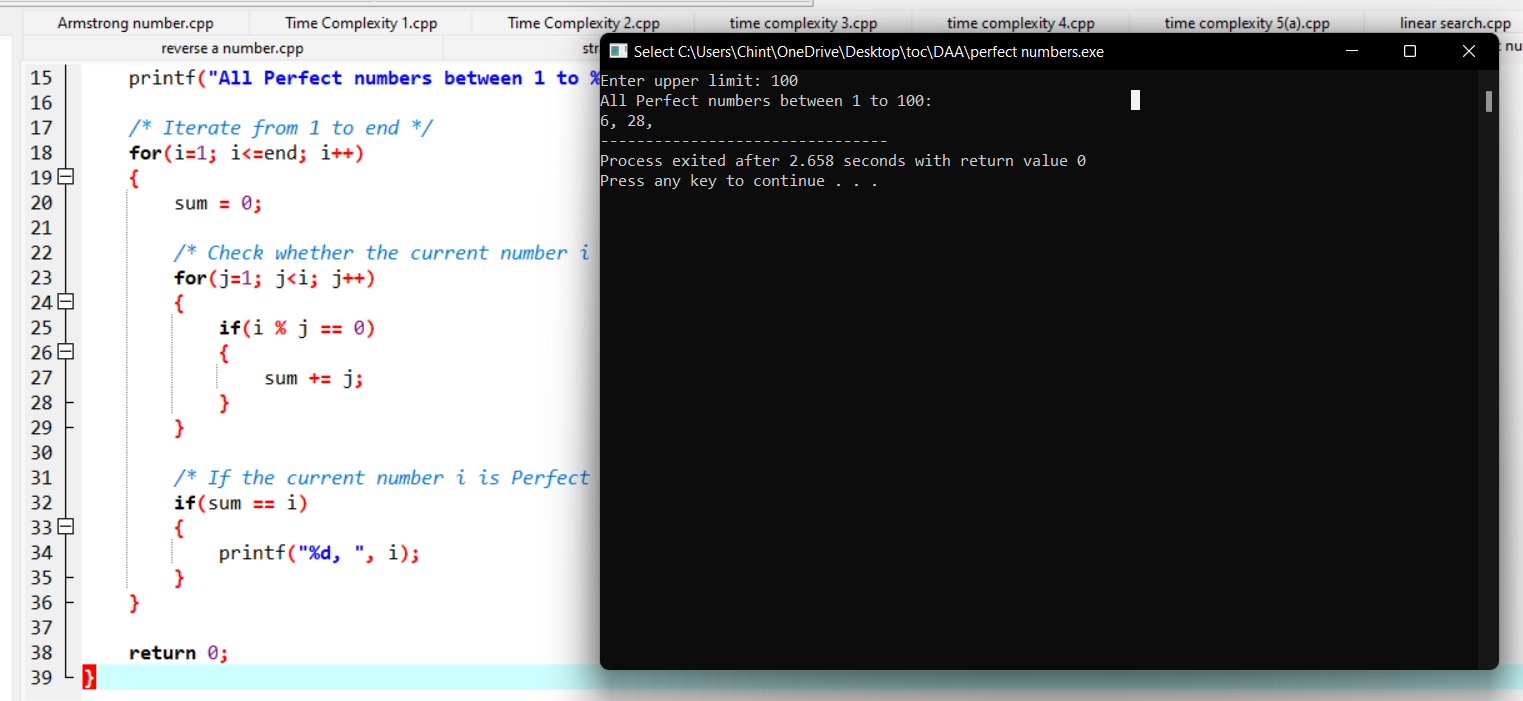
}

}

return 0;

}

**Output:**



**20.Program to Find Even Sum of Fibonacci Series Till number N?(day 2)**

**Sample Input: n = 4**

**Sample Output: 33**

**(N = 4, So here the Fibonacci series will be produced from 0th term till 8th term: 0, 1, 1, 2, 3, 5, 8, 13, 21**

**Sum of numbers at even indexes = 0 + 1 + 3 + 8 + 21 = 33)**

# Program:

#include<stdio.h>

int fab(int n){

int n1=0,n2=1,n3,c=0,sum=0,ini=0;

int count=0;

printf("%d %d ",n1,n2);

while(c<n){

count++;

n3=n1+n2;

count++;

printf("%d ",n3);

if(ini%2==0){

count++;

sum+=n3;

count++;

c++;

count++;

}

ini++;

count++;

n1=n2;

count++;

n2=n3;

count++;

}count++;

printf("\nsum: %d\n",sum);

printf("Time complexity: %d",count);

return 0;

}

int main(){

int n;

printf("Enter the n value: ");

scanf("%d",&n);

fab(n);

}

# Output:

Enter the n value: 4

0 1 1 2 3 5 8 13 21

sum: 33

Time complexity: 48

--------------------------------

**21.Write a program to perform Selection sort and estimate time Complexity**

**Estimate the time iteration for the following set of numbers.**

1. **(10,5, 80,-2,5,23, 45) B. (12, 3, 0, 34, -11, 34, 22, 8) C.( 3, 35, -56, 66, 77, ,-78, 82)**

# Program:

#include <stdio.h>

void selectionSort(int arr[], int size);

void swap(int \*a, int \*b);

void selectionSort(int arr[], int size)

{

int i, j;

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

{

for (j = i ; j < size; j++)

{

if (arr[i] > arr[j])

swap(&arr[i], &arr[j]);

}

}

}

void swap(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

int main()

{

int array[10], i, size;

printf("How many numbers you want to sort: ");

scanf("%d", &size);

printf("\nEnter %d numbers\t", size);

printf("\n");

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

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

selectionSort(array, size);

printf("\nSorted array is ");

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

printf(" %d ", array[i]);

return 0;

}

# Output:

How many numbers you want to sort: 5

Enter 5 numbers

5

6

1

4

2

Sorted array is 1 2 4 5 6

--------------------------------

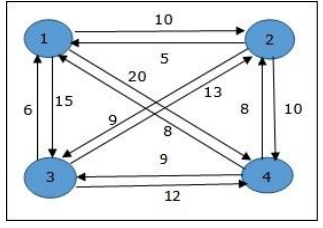
**22. Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers. The**

**inputs are weighted, directed graph, and n, the number of vertices in the graph. The graph is**

**represented by a two-dimensional array W, which has both its rows and columns indexed from 1 to n,**

**where W [i] [j] is the weight on the edge from the ith vertex to the jth vertex. Write a program for**

**travelling salesman problem using dynamic programming for the below given graph.**

****

**Program:**

#include <iostream>

using namespace std;

const int n = 4;

const int MAX = 1000000;

int dist[n + 1][n + 1] = {

{ 0, 0, 0, 0, 0 }, { 0, 0, 10, 15, 20 },

{ 0, 10, 0, 25, 25 }, { 0, 15, 25, 0, 30 },

{ 0, 20, 25, 30, 0 },

};

int memo[n + 1][1 << (n + 1)];

int fun(int i, int mask)

{

if (mask == ((1 << i) | 3))

return dist[1][i];

if (memo[i][mask] != 0)

return memo[i][mask];

int res = MAX;

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

if ((mask & (1 << j)) && j != i && j != 1)

res = std::min(res, fun(j, mask & (~(1 << i)))

+ dist[j][i]);

return memo[i][mask] = res;

}

int main()

{

int ans = MAX;

for (int i = 1; i <= n; i++)

ans = std::min(ans, fun(i, (1 << (n + 1)) - 1)

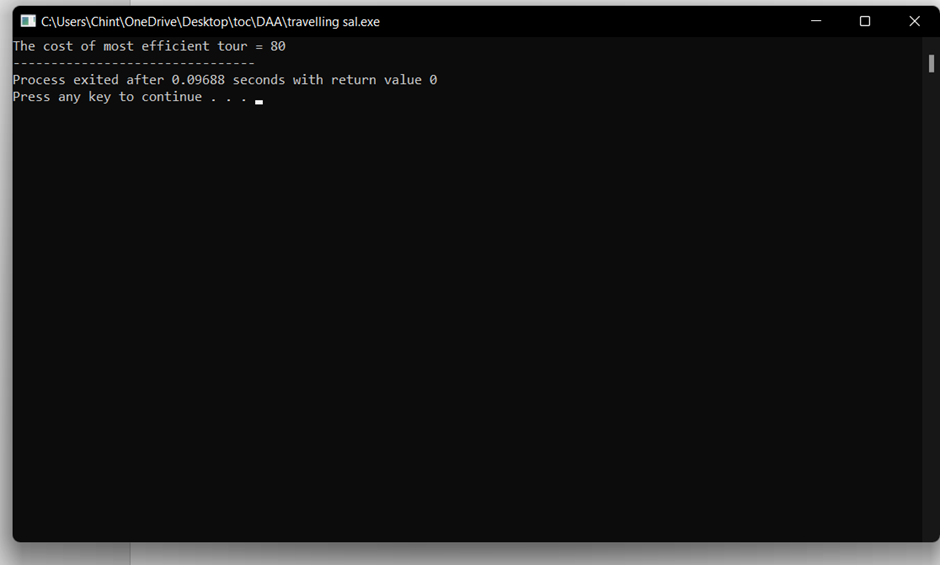
+ dist[i][1]);

printf("The cost of most efficient tour = %d", ans);

return 0;

}

**Output:**

****

**23. Write a program using choice to check**

**Case 1: Given string is palindrome or not**

**Case 2: Given number is palindrome or not**

**Sample Input:**

**Case = 1**

**String = MADAM**

**Sample Output:**

**Palindrome**

**Test cases:**

1. **MONEY**
2. **5678765**
3. **MALAY12321ALAM**
4. **MALAYALAM**
5. **1234.4321**

**24.Write a program to inset a number in a list**

**Testing Condition**

* + 1. **Insert at the beginning**
    2. **Insert in the middle**
    3. **Insert at the last**
    4. **Not Available position in a list**

# Program:

#include<stdio.h>

#include<stdlib.h>

#define MAX 100

int list[MAX];

int position, num, size = 0;

void insertAtBeginning() {

int i;

if (size == MAX) {

printf("List is full. Cannot insert at the beginning\n");

return;

}

for (i = size; i > 0; i--) {

list[i] = list[i - 1];

}

list[0] = num;

size++;

printf("Number %d inserted at position %d\n", num, position);

}

void insertInMiddle() {

int i;

if (size == MAX) {

printf("List is full. Cannot insert in the middle\n");

return;

}

for (i = size; i > position - 1; i--) {

list[i] = list[i - 1];

}

list[position - 1] = num;

size++;

printf("Number %d inserted at position %d\n", num, position);

}

void insertAtLast() {

if (size == MAX) {

printf("List is full. Cannot insert at the end\n");

return;

}

list[size] = num;

size++;

printf("Number %d inserted at position %d\n", num, size);

}

int main() {

int choice;

while (1) {

printf("1. Insert at the beginning\n");

printf("2. Insert in the middle\n");

printf("3. Insert at the last\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the number to insert: ");

scanf("%d", &num);

position = 1;

insertAtBeginning();

break;

case 2:

printf("Enter the number to insert: ");

scanf("%d", &num);

printf("Enter the position to insert the number: ");

scanf("%d", &position);

if (position < 1 || position > size + 1) {

printf("Invalid position\n");

} else {

insertInMiddle();

}

break;

case 3:

printf("Enter the number to insert: ");

scanf("%d", &num);

position = size + 1;

insertAtLast();

break;

case 4:

exit(0);

default:

printf("Invalid choice\n");

break;

}

}

return 0;

}

# Output:

Insert at the beginning

2. Insert in the middle

3. Insert at the last

4. Exit

Enter your choice: 1

Enter the number to insert: 5

Number 5 inserted at position 1

1. Insert at the beginning

2. Insert in the middle

3. Insert at the last

4. Exit

Enter your choice: 2

Enter the number to insert: 6

Enter the position to insert the number: 2

Number 6 inserted at position 2

1. Insert at the beginning

2. Insert in the middle

3. Insert at the last

4. Exit

Enter your choice: 3

Enter the number to insert: 4

Number 4 inserted at position 3

1. Insert at the beginning

2. Insert in the middle

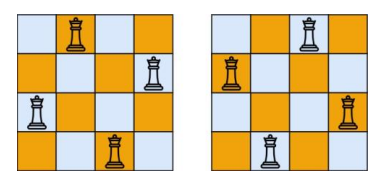
3. Insert at the last

4. Exit

Enter your choice: 4

------------------------

**25.The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Write a program for the same.**

****

**Program:**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int a[30],count=0;

int place(int pos) {

int i;

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

if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))

return 0;

}

return 1;

}

void print\_sol(int n) {

int i,j;

count++;

printf("\n\ntime comolexity #%d:\n",count);

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

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

if(a[i]==j)

printf("Q\t"); else

printf("\*\t");

}

printf("\n");

}

}

void queen(int n) {

int k=1;

a[k]=0;

while(k!=0) {

a[k]=a[k]+1;

while((a[k]<=n)&&!place(k))

a[k]++;

if(a[k]<=n) {

if(k==n)

print\_sol(n); else {

k++;

a[k]=0;

}

} else

k--;

}

}

main() {

int i,n;

printf("Enter the number of Queens\n");

scanf("%d",&n);

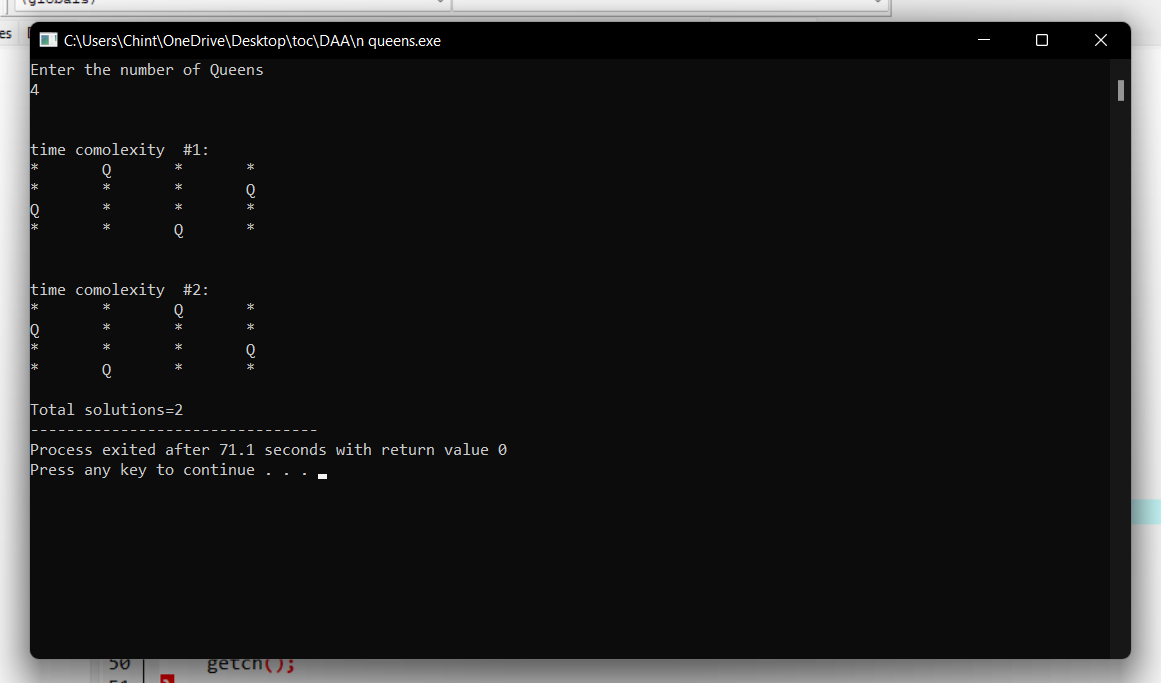
queen(n);

printf("\nTotal solutions=%d",count);

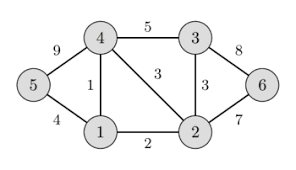
getch();

**}**

**Output:**



**26.Write a program to perform Minimum spanning tree using greedy techniques and estimate time complexity for the given set of values.**

****

# Program:

#include <limits.h>

#include <stdbool.h>

#include <stdio.h>

#define V 5

int minKey(int key[], bool mstSet[])

{

int min = INT\_MAX, min\_index,count=0;

for (int v = 0; v < V; v++)

if (mstSet[v] == false && key[v] < min)

min = key[v], min\_index = v;

count++;

return min\_index;

}

int printMST(int parent[], int graph[V][V])

{

int count=0;

printf("Edge \tWeight\n");

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

printf("%d - %d \t%d \n", parent[i], i,

graph[i][parent[i]]);

count++;

}

void primMST(int graph[V][V])

{

int count=0;

int parent[V];

int key[V];

bool mstSet[V];

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

key[i] = INT\_MAX, mstSet[i] = false;

count++;

key[0] = 0;

count++;

parent[0] = -1;

count++;

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

int u = minKey(key, mstSet);

count++;

mstSet[u] = true;

count++;

for (int v = 0; v < V; v++)

if (graph[u][v] && mstSet[v] == false

&& graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

}

printMST(parent, graph);

printf(" time completity is %d",count);

}

int main()

{

int graph[V][V] = { { 0, 2, 0, 6, 0 },

{ 2, 0, 3, 8, 5 },

{ 0, 3, 0, 0, 7 },

{ 6, 8, 0, 0, 9 },

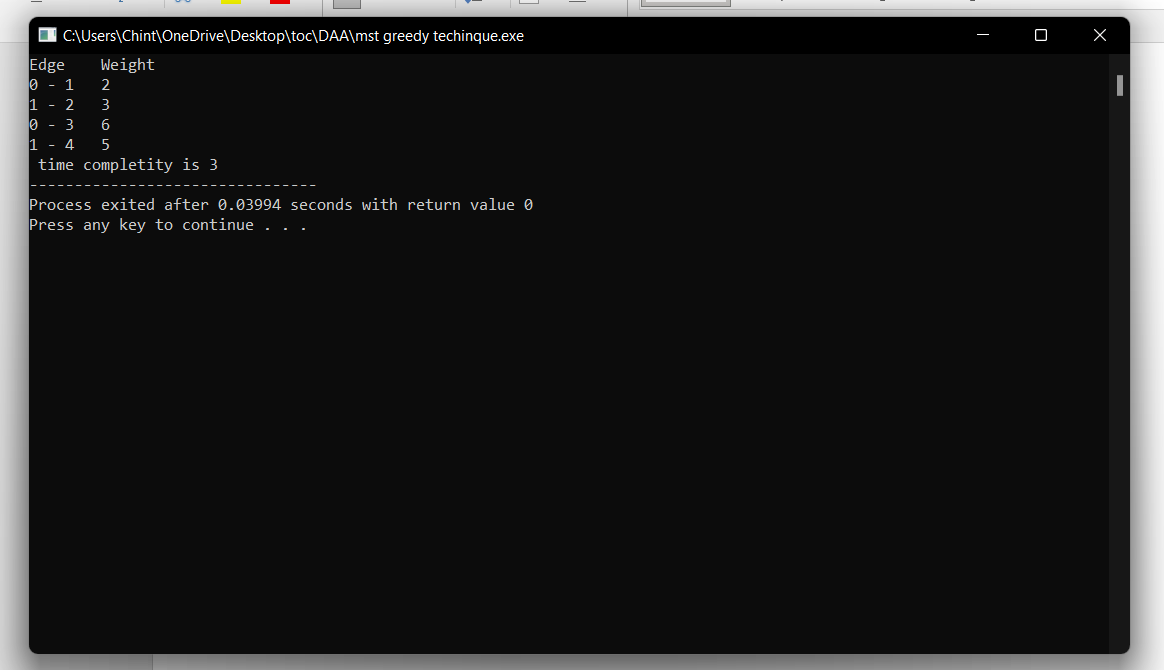
{ 0, 5, 7, 9, 0 } };

primMST(graph);

return 0;

}

# Output:



1. **Write a program to perform sorting without using swapping and estimate time complexity.**

# Program:

#include <stdio.h>

void selection\_sort(int arr[], int n) {

int i, j, min\_idx;

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

min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

int temp = arr[min\_idx];

for (j = min\_idx; j > i; j--)

arr[j] = arr[j-1];

arr[i] = temp;

}

}

int main() {

int arr[] = {64, 25, 12, 22, 11};

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

selection\_sort(arr, n);

printf("Sorted array: \n");

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

printf("%d ", arr[i]);

return 0;

}

# Output:

Sorted array:

11 12 22 25 64

-------------------

1. **Write a program to perform Bubble sort and estimate time Complexity for n values.**

**Perform test cases for the following set of numbers.**

**A..(10,5, 80,-2,5,23, 45) B. (12, 3, 0, 34, -11, 34, 22, 8) C.( 3, 35, -56, 66, 77, ,-78, 82)**

# Program:

#include<stdio.h>

int main(){

int ele,count=0;

printf("Enter total element: ");

scanf("%d",&ele);

int arr[ele];

printf("Enter the elements: ");

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

count++;

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

}count++;

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

{

count++;

for (int j =i+1; j < ele; j++)

{

count++;

if (arr[i]>arr[j])

{

count++;

int temp=arr[i];

count++;

arr[i]=arr[j];

count++;

arr[j]=temp;

count++;

}

}count++;

}count++;

printf("sorted array: ");

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

{count++;

count++;

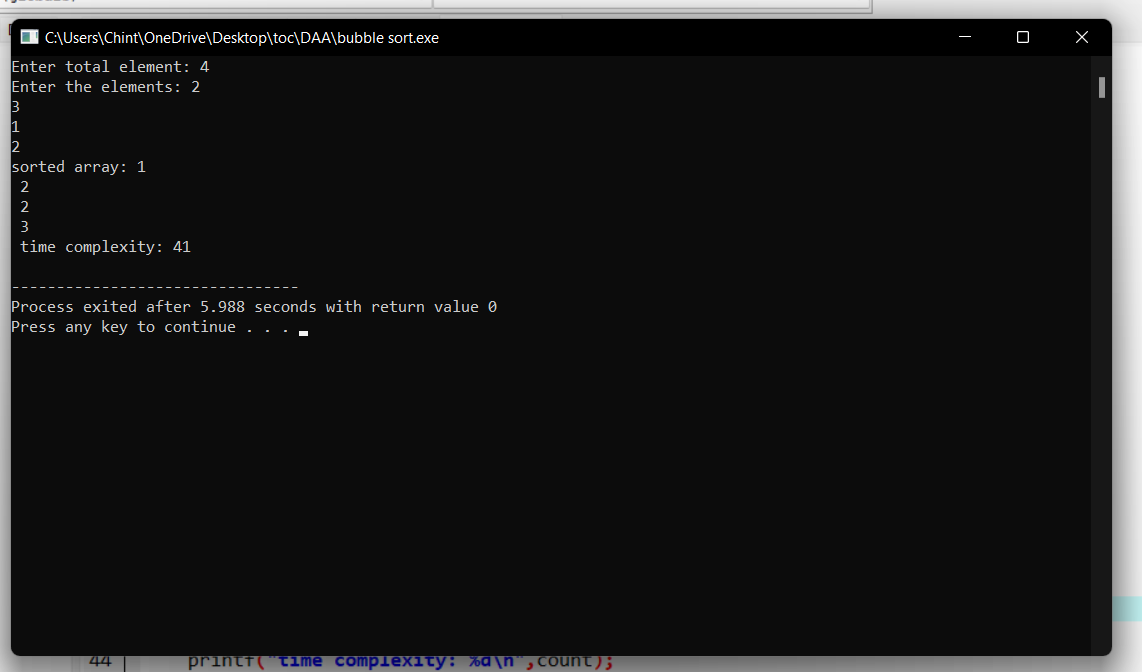
printf("%d \n ",arr[i]);

}count++;

printf("time complexity: %d\n",count);

}

OUTPUT:



**29.Write a program to print the reverse of a string. And estimate the time complexity**

**for the given inputs.**

**Test cases: output –**

**“ as\nr5Y” Y5rn|sa**

**“7yut02” 20tuy7**

**“EryEq qEyrE**

**Program:**

#include<stdio.h>

int main(){

char val[25];

printf("enter the value: ");

scanf("%s",&val);

int count=0,c=0;

while (val[count]!='\0'){

count++;

c++;

}c++;

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

c++;

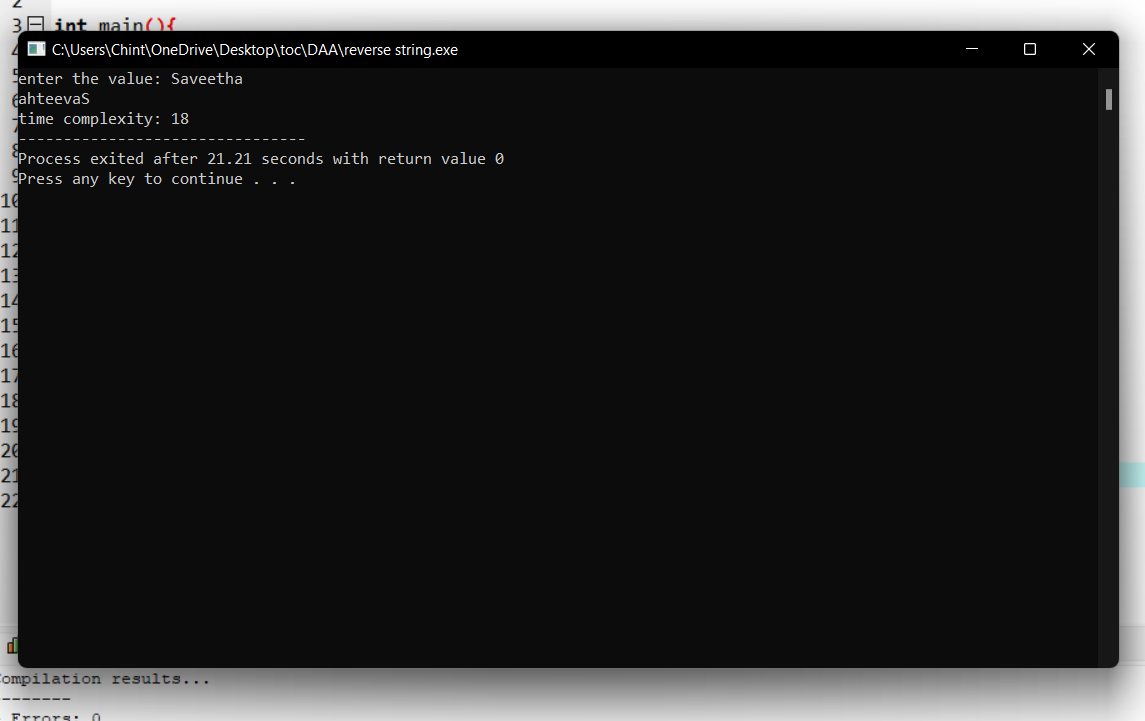
printf("%c",val[i]);

}c++;

printf("\ntime complexity: %d",c);

}

**Output:**



**30.Write a program using dynamic programming to find out the optimal binary**

**search tree for the given input**

**P : successful search q:Unsuccessful Search**

**P(1:4) = (2,3,1,1) q(0:4) = (2,3,1,1,2)**

# Program:

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define MAX\_NODES 100

#define MIN(a, b) ((a) < (b) ? (a) : (b))

int keys[MAX\_NODES]; // array of keys

int freq[MAX\_NODES]; // frequency of each key

int cost[MAX\_NODES][MAX\_NODES]; // cost matrix

int sum[MAX\_NODES]; // cumulative frequency

int dp[MAX\_NODES][MAX\_NODES]; // dp array to store subproblems

int n; // number of nodes

int optimalSearchTree(int i, int j) {

if (dp[i][j] != -1) {

return dp[i][j];

}

if (i == j) {

return dp[i][j] = freq[i];

}

int minCost = INT\_MAX;

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

int c = optimalSearchTree(i, r - 1) + optimalSearchTree(r + 1, j) + sum[j] - sum[i - 1];

minCost = MIN(minCost, c);

}

return dp[i][j] = minCost;

}

int main() {

printf("Enter the number of nodes: ");

scanf("%d", &n);

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

printf("Enter key[%d] and its frequency: ", i);

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

}

// calculate cumulative frequency

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

sum[i] = sum[i - 1] + freq[i];

}

// initialize dp array

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

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

dp[i][j] = -1;

}

}

printf("The minimum cost of the optimal binary search tree is: %d\n", optimalSearchTree(0, n - 1));

return 0;

}

# Output:

Enter the number of nodes: 4

Enter key[0] and its frequency: 2

1

Enter key[1] and its frequency: 3

1

Enter key[2] and its frequency: 1

4

Enter key[3] and its frequency: 1

4

The minimum cost of the optimal binary search tree is: -2147483632

--------------------------------

**31. Write a program to perform permutation of an array of integers and make all the**

**arrangement are to be in possible sequence.**

**Input a{]={1,2,3) Output [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].**

# Program:

#include <stdio.h>

#include <stdlib.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void permute(int \*arr, int start, int end) {

if (start == end) {

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

printf("%d ", arr[i]);

printf("\n");

} else {

for (int i = start; i <= end; i++) {

swap(&arr[start], &arr[i]);

permute(arr, start + 1, end);

swap(&arr[start], &arr[i]);

}

}

}

int main() {

int n;

printf("Enter the number of elements in the array: ");

scanf("%d", &n);

int arr[n];

printf("Enter the elements of the array: ");

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

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

printf("The permutations are:\n");

permute(arr, 0, n - 1);

return 0;

}

# Output:

Enter the number of elements in the array: 4

Enter the elements of the array: 8

7

3

2

The permutations are:

8 7 3 2

8 7 2 3

8 3 7 2

8 3 2 7

8 2 3 7

8 2 7 3

7 8 3 2

7 8 2 3

7 3 8 2

7 3 2 8

7 2 3 8

7 2 8 3

3 7 8 2

3 7 2 8

3 8 7 2

3 8 2 7

3 2 8 7

3 2 7 8

2 7 3 8

2 7 8 3

2 3 7 8

2 3 8 7

2 8 3 7

2 8 7 3

--------------------------------

**32. Write a program to check the given no is palindrome or not**

**Given an integer x, return true if x is a palindrome, and false otherwise**

**input out put**

**121 True**

**234 False**

1. **True**

**Program:**

#include<stdio.h>

int main()

{

int i,n,r,s=0;

printf("\n Enter Integer Number:");

scanf("%d",&n);

//LOOP TO FIND REVERSE OF A NUMBER

for(i=n;i>0; )

{

r=i%10;

s=s\*10+r;

i=i/10;

}

/\* CHECKING IF THE NUMBER ENTERED AND THE REVERSE NUMBER IS EQUAL OR NOT \*/

if(s==n)

{

printf("\n %d is a Palindrome Number",n);

}

else

{

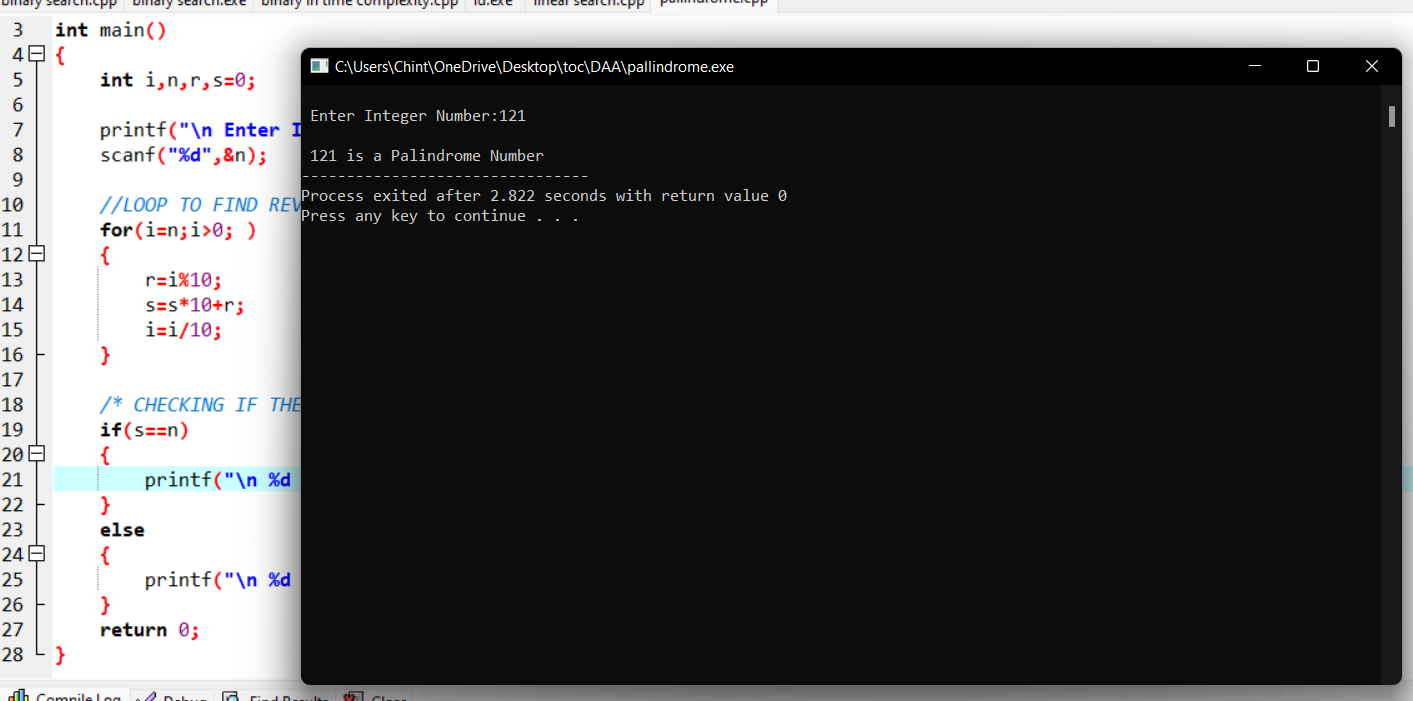
printf("\n %d is not a Palindrome Number",n);

}

return 0;

}

**Output:**



**33.Write a program for the given pattern the given pattern If n=4**

**1**

1 2

1 2 3

1. 2 3 4

# Program:

#include<stdio.h>

int main()

{

int rows, i, j;

int count=0;

printf("Enter the number of rows: ");

scanf("%d",&rows);

for(i = 1; i <= rows; i++)

{

for(j = rows; j > i; j--)

{

printf(" ");

count++;

}

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

{

printf("%d ",j);

count++;

}

printf("\n");

}

printf("Time complexity:%d",count);

return 0;

}

# Output:

Enter the number of rows: 4

1

1 2

1 2 3

1 2 3 4

Time complexity:16v

**34.Write a program to find out Hamiltonian circuit Using backtracking method.**

**And estimate the time complexity for the given set of elements is**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **a** | **b** | **c** | **d** | **e** | **f** |
| **a** | **0** | **0** | **1** | **1** | **1** | **1** |
| **b** | **0** | **0** | **1** | **0** | **0** | **1** |
| **c** | **1** | **1** | **0** | **1** | **1** | **1** |
| **d** | **1** | **0** | **1** | **0** | **1** | **0** |
| **e** | **1** | **0** | **0** | **1** | **0** | **0** |
| **f** | **1** | **1** | **1** | **0** | **0** | **0** |

# Program:

#include <stdio.h>

#define MAX 20

int n;

int adj[MAX][MAX];

int x[MAX];

int is\_safe(int v, int pos) {

int i;

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

if (adj[v][i] && x[i] == pos)

return 0;

return 1;

}

int hamiltonian\_cycle(int pos) {

int v;

if (pos == n) {

if (adj[x[pos - 1]][x[0]])

return 1;

else

return 0;

}

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

if (is\_safe(v, pos)) {

x[pos] = v;

if (hamiltonian\_cycle(pos + 1))

return 1;

x[pos] = -1;

}

}

return 0;

}

int main() {

int i, j;

printf("Enter number of vertices: ");

scanf("%d", &n);

printf("Enter adjacency matrix:\n");

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

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

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

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

x[i] = -1;

x[0] = 0;

if (hamiltonian\_cycle(1) == 0)

printf("\nSolution does not exist\n");

else {

printf("\nSolution exists:\n");

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

printf("%d ", x[i]);

printf("%d", x[0]);

}

return 0;

}

Output:

Enter number of vertices: 3

Enter adjacency matrix:

0

2

3

4

0

6

9

8

0

Solution exists:

0 0 1 0

--------------------------------

**35.Write a program to return all the possible subsets for a given integer array. Return the**

**solution in any order.**

**Input nums= [1,2,3]**

**Output : [ [], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]]**

**Program:**

**36.Write a program to compute container loader Problem for the given values and estimate time complexity.**

**N=8 be total no of containers having weights (w1, w2, w3,…w8) = [ 50, 100, 30, 80, 90, 200, 150, 20 ]. Capacity value = 100**

**Program:**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_ITEMS 100

#define MAX\_WEIGHT 100

int weight[MAX\_ITEMS];

int value[MAX\_ITEMS];

int dp[MAX\_ITEMS][MAX\_WEIGHT];

int max(int a, int b) {

return (a > b) ? a : b;

}

int knapsack(int n, int w) {

int i, j;

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

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

if (i == 0 || j == 0) {

dp[i][j] = 0;

} else if (weight[i-1] <= j) {

dp[i][j] = max(value[i-1] + dp[i-1][j-weight[i-1]], dp[i-1][j]);

} else {

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

}

}

}

return dp[n][w];

}

int main() {

int n = 4;

int w = 10;

weight[0] = 1;

weight[1] = 2;

weight[2] = 3;

weight[4] = 4;

weight[5] = 5;

weight[6] = 6;

weight[7] = 7;

weight[8] = 8;

value[0] = 50;

value[1] = 100;

value[2] = 30;

value[3] = 80;

value[4] = 90;

value[5] = 200;

value[6] = 150;

value[7] = 20;

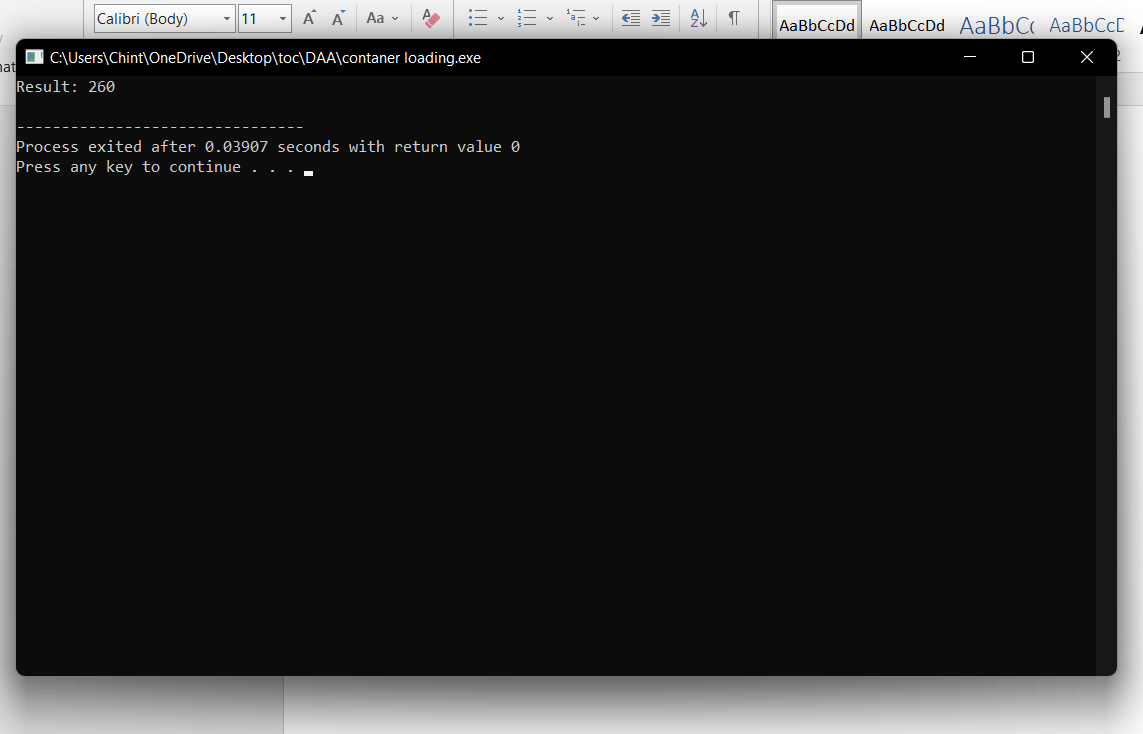
int result = knapsack(n, w);

printf("Result: %d\n", result);

return 0;

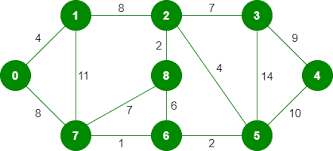
}

**Output:**



**37.Write a program to find a minimum spanning tree using prims technique for the**

**given graph.**

****

**Program:**

#include <stdio.h>

#include <limits.h>

#define vertices 5

int minimum\_key(int k[], int mst[])

{

int minimum = INT\_MAX, min,i,count=0;

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

if (mst[i] == 0 && k[i] < minimum )

minimum = k[i], min = i;

return min;

count++;

}

void prim(int g[vertices][vertices])

{

int parent[vertices];

int k[vertices];

int mst[vertices];

int i, count,edge,v;

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

{

k[i] = INT\_MAX;

count++;

mst[i] = 0;

count++;

}

count++;

k[0] = 0;

count++;

parent[0] = -1;

count++;

for (count = 0; count < vertices-1; count++)

{

edge = minimum\_key(k, mst);

mst[edge] = 1;

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

{

if (g[edge][v] && mst[v] == 0 && g[edge][v] < k[v])

{

parent[v] = edge, k[v] = g[edge][v];

}

}

}

count++;

count++;

count++;

printf("\n Edge \t Weight\n");

for (i = 1; i < vertices; i++)

printf(" %d <-> %d %d \n", parent[i], i, g[i][parent[i]]);

count++;

printf(" time complexity is :%d",count);

}

int main()

{

int g[vertices][vertices] = {{0, 0, 3, 0, 0},

{0, 0, 10, 4, 0},

{3, 10, 0, 2, 6},

{0, 4, 2, 0, 1},

{0, 0, 6, 1, 0},

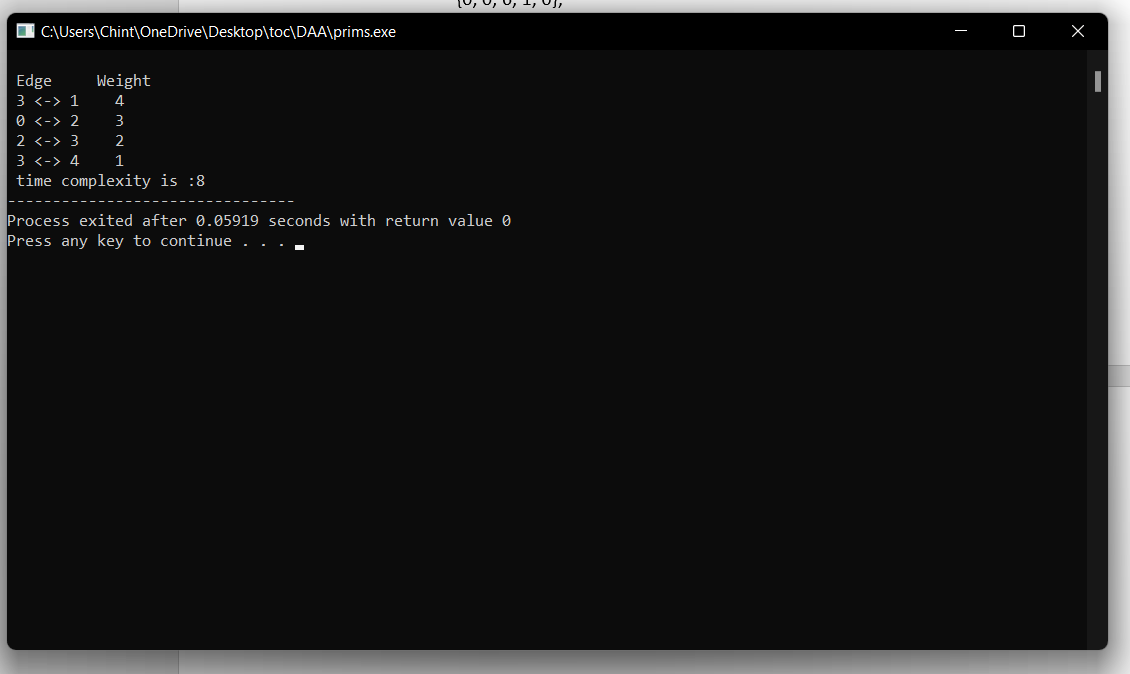
};

prim(g);

return 0;

}

**Output:**



**38.Write a program to print a minimum and maximum value sequency for all the numbers in a list.**

**Input a[]=(3, 5, -4, 1, 8, 2, 0, 4) Output (-4, 8, 0, 5, 1, 4, 3, 2)**

**39.Write a program to check sub string is there in a string or not.**

**Input/Output**

**a.orginal string = "babad" b.Orginal string = "babad" c. Orginal string = "babad"**

**Sub string = "shahad" Sub string = "daa" Sub string = "aba"**

**Output = Found Output = Not Found Output = Found**

**Program:**

#include<stdio.h>

int main()

{

char str[80], search[10];

int count1 = 0, count2 = 0, i, j, flag;

printf("Enter a string:");

gets(str);

printf("Enter search substring:");

gets(search);

while (str[count1] != '\0')

count1++;

while (search[count2] != '\0')

count2++;

for (i = 0; i <= count1 - count2; i++)

{

for (j = i; j < i + count2; j++)

{

flag = 1;

if (str[j] != search[j - i])

{

flag = 0;

break;

}

}

if (flag == 1)

break;

}

if (flag == 1)

printf("SEARCH SUCCESSFUL!");

else

printf("SEARCH UNSUCCESSFUL!");

return 0;

}

**Output:**

