**1)linear search**

#include<stdio.h>

int main(){

int a[]={1,2,3,4,5};

int flag,key;

printf("enter the key:");

scanf("%d",&key);

for(int i=0;i<7;i++){

if(a[i]==key){

flag=1;

break;

}

else{

flag=0;

}

}

if(flag==1){

printf("the key is found ");

}

else{

printf("the key is not found");

}

return 0;

}

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

**2)binary search**

#include <stdio.h>

int binarySearch(int arr[], int left, int right, int target) {

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target)

return mid;

if (arr[mid] < target)

left = mid + 1;

else

right = mid - 1;

}

return -1;

}

int main() {

int arr[] = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};

int target = 12;

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

int result = binarySearch(arr, 0, n - 1, target);

if (result == -1)

printf("Element is not present in array\n");

else

printf("Element is present at index %d\n", result);

return 0;

}

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

**3)bubble sort**

#include<stdio.h>

int bubblesort(int a[],int n);

int i,j;

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

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

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

int temp=a[j+1];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

int main(){

int a[]={1,2,3,4,5,67,8,9,10};

int n=sizeof(a[]);

bubblesort(a[],n);

printf("sorted array:");

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

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

return 0;

}

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

**4)selection sort**

#include <stdio.h>

void selectionSort(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];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

int main() {

int arr[] = {64, 34, 25, 12, 22, 11, 90};

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

selectionSort(arr, n);

printf("Sorted array: ");

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

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

return 0;

}

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

**5)binomial coefficient**

#include <stdio.h>

int bin(int n) {

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

int num = 1;

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

printf(" ");

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

printf(" ");

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

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

printf(" %d", num);

}

printf("\n");

}

}

int main() {

int n ;

printf("value of n:");

scanf("%d",&n);

bin(n);

   return 0;

}

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

**6) insertion sort**

#include <math.h>

#include <stdio.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

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

{

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key)

{

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

j = j - 1;

}

arr[j + 1] = key;

}

}

void printArray(int arr[], int n)

{

int i;

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

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

printf("\n");

}

int main()

{

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

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

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

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

**7)quick sort**

#include <stdio.h>

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

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

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

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void printArray(int arr[], int size) {

int i;

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

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

printf("\n");

}

int main() {

int arr[] = { 12, 17, 6, 25, 1, 5 };

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

quickSort(arr, 0, n - 1);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

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

**8)merge sort**

#include <stdio.h>

int main()

{

int arr[100],temp[100],i,j,k,n,size,l1,h1,l2,h2;

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

scanf("%d",&n);

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

{

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

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

}

printf("Unsorted list is : ");

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

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

for(size=1; size < n; size=size\*2 )

{

l1=0;

k=0;

while( l1+size < n)

{

h1=l1+size-1;

l2=h1+1;

h2=l2+size-1;

if( h2>=n )

h2=n-1;

i=l1;

j=l2;

while(i<=h1 && j<=h2 )

{

if( arr[i] <= arr[j] )

temp[k++]=arr[i++];

else

temp[k++]=arr[j++];

}

while(i<=h1)

temp[k++]=arr[i++];

while(j<=h2)

temp[k++]=arr[j++];

l1=h2+1;

}

for(i=l1; k<n; i++)

temp[k++]=arr[i];

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

arr[i]=temp[i];

printf("\nSize=%d \nElements are :",size);

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

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

}

printf("Sorted list is :\n");

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

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

printf("\n");

return 0;

}

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

**9)knapsack problem**

#include <stdio.h>

int max(int a, int b) { return (a > b) ? a : b; }

int knapSack(int W, int wt[], int val[], int n)

{

if (n == 0 || W == 0)

return 0;

if (wt[n - 1] > W)

return knapSack(W, wt, val, n - 1);

else

return max( val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),

knapSack(W, wt, val, n - 1));

}

int main()

{

int profit[] = { 60, 100, 120 };

int weight[] = { 10, 20, 30 };

int W = 50;

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

printf("%d", knapSack(W, weight, profit, n));

    return 0;

}

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

**10) tsp**

**----------**---------------------------------------------------------------------------------------------------------------------------------------------------------------

**11)revese a number**

#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;

}

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

**12)strassens matrix multiplication**

#include<stdio.h>

int main(){

int a[2][2],b[2][2],c[2][2],i,j;

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

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

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

printf("\n");

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

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

}

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

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

printf("\n");

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

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

}

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

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

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

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

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

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

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

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

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

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

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

printf("\nAfter multiplication using \n");C

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

printf("\n");

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

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

}

return 0;

}

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

**13)Armstrong number**

#include<stdio.h>

int main()

{

int n,r,sum=0,temp;

printf("enter the number=");

scanf("%d",&n);

temp=n;

while(n>0)

{

r=n%10;

sum=sum+(r\*r\*r);

n=n/10;

}

if(temp==sum)

printf("armstrong number ");

else

printf("not armstrong number");

return 0;

}

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

**14) factorial number**

#include <stdio.h>

int factorial( int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

int main() {

int number;

printf("Enter a integer: ");

scanf("%d", &number);

int result = factorial(number);

printf("Factorial of %d is %d\n", number, result);

return 0;

}

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

**15)sum of digits**

#include<stdio.h>

int main(){

int n,sum,num;

printf("enter the digits:");

scanf("%d",&n);

while(n!=0){

sum=sum+(n%10);

num=num/10;

}

printf("%d",sum);

return 0;

}

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

**16)perfect no**

#include <stdio.h>

int isPerfectNumber(int num) {

int sum = 0;

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (isPerfectNumber(number)) {

printf("%d is a perfect number.\n", number);

} else {

printf("%d is not a perfect number.\n", number);

}

return 0;

}

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

**17)fibanocci**

#include<stdio.h>

int main()

{

int n1=0,n2=1,n3,i,number;

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

scanf("%d",&number);

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

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

{

n3=n1+n2;

printf(" %d",n3);

n1=n2;

n2=n3;

}

return 0;

}

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

**18)gcd and lcm**

#include<stdio.h>

int main(){

int n1,n2,gcd,i;

printf("enter the numbers:");

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

for(i=1;i<=n1||i<=n2;i++){

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

gcd=i;

}

}

printf("the gcd of %d and %d is %d",n1,n2,gcd);

return 0;

}

Lcm:

#include<stdio.h>

int main()

{

int a, b, max;

printf("Enter the two numbers : ");

scanf("%d %d", &a, &b);

max = (a > b) ? a : b;

while (max % a != 0 || max % b != 0)

{

max++;

}

printf("The LCM of %d and %d is %d.", a, b, max);

return 0;

}

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

**19)palindrome**

#include <stdio.h>

int main() {

int n, reversed = 0, remainder, original;

printf("Enter an integer: ");

scanf("%d", &n);

original = n;

while (n != 0) {

remainder = n % 10;

reversed = reversed \* 10 + remainder;

n /= 10;

}

if (original == reversed)

printf("%d is a palindrome.", original);

else

printf("%d is not a palindrome.", original);

return 0;

}

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

**20)mth maximum and nth minimum**

#include <stdio.h>

#include <conio.h>

int main()

{

int a[1000],i,n,min,max;

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

scanf("%d",&n);

printf("Enter elements in array : ");

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

{

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

}

printf("mth minimum of array is : ");

printf("nth maximum of array is : ");

min=max=a[0];

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

{

if(min>a[i])

min=a[i];

if(max<a[i])

max=a[i];

}

printf("minimum of array is : %d",min);

printf("\nmaximum of array is : %d",max);

return 0;

}

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

**21)Hamiltonian circuit**

#include<stdio.h>

#define V 5

void printSolution(int path[]);

bool isSafe(int v, bool graph[V][V], int path[], 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 hamCycleUtil(bool graph[V][V], int path[], int pos)

{

if (pos == V)

{

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

return true;

else

return false;

}

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

{

if (isSafe(v, graph, path, pos))

{

path[pos] = v;

if (hamCycleUtil (graph, path, pos+1) == true)

return true;

path[pos] = -1;

}

}

return false;

}

bool hamCycle(bool graph[V][V])

{

int \*path = new int[V];

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

path[i] = -1;

path[0] = 0;

if ( hamCycleUtil(graph, path, 1) == false )

{

printf("\nSolution does not exist");

return false;

}

printSolution(path);

return true;

}

void printSolution(int path[])

{

printf ("Solution Exists:"

" Following is one Hamiltonian Cycle \n");

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

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

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

printf("\n");

}

int main()

{

bool graph1[V][V] = {{0, 1, 0, 1, 0},

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

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

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

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

};

hamCycle(graph1);

bool graph2[V][V] = {{0, 1, 0, 1, 0},

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

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

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

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

};

hamCycle(graph2);

return 0;

}

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

**22)flyods algorithm**#include<stdio.h>

void floyd(int a[4][4], int n)

{

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

{

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

{

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

{

if(a[i][j]>a[i][k]+a[k][j])

{

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

}

}

}

}

printf("All Pairs Shortest Path is :\n");

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

{

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

{

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

}

printf("\n");

}

}

int main()

{

int cost[4][4] = {{0, 3, 999, 4}, {8, 0, 2, 999}, {5, 999, 0, 1}, {2, 999, 999, 0}};

int n = 4;

floyd(cost,n);

}

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

**23)insert a number**

#include <stdio.h>

int main() {

int arr[100];

int size, i, position, element;

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

scanf("%d", &size);

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

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

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

}

printf("Enter the element to insert at the beginning: ");

scanf("%d", &element);

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

arr[i] = arr[i - 1];

}

arr[0] = element;

size++;

printf("\nEnter the position to insert at the middle: ");

scanf("%d", &position);

if (position < 0 || position > size) {

printf("\nInvalid position.\n");

return 1;

}

printf("\nEnter the element to insert at the middle: ");

scanf("%d", &element);

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

arr[i] = arr[i - 1];

}

arr[position] = element;

size++;

printf("\nEnter the element to insert at the end: ");

scanf("%d", &element);

arr[size] = element;

size++;

printf("Array after insertion at the beginning:\n");

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

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

}

return 0;

}

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

**24) sum of subsets**

#include <stdio.h>

int isSubsetSum(int set[], int n, int sum)

    if (sum == 0) return 1;

    if (n == 0) return 0;

    if (set[n-1] > sum) return isSubsetSum(set, n-1, sum);

    return isSubsetSum(set, n-1, sum) || isSubsetSum(set, n-1, sum - set[n-1]);

}

int main() {

    int set[] = {3, 34, 4, 12, 5, 2};

    int sum = 9;

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

    if (isSubsetSum(set, n, sum)) printf("Found a subset with given sum");

    else printf("No subset with given sum");

    return 0;

}

**25)min and max using divide and conquer**