**1) 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 val[] = {60, 100, 120};

    int wt[] = {10, 20, 30};

    int  W = 50;

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

    printf("%d", knapSack(W, wt, val, n));

    return 0;

}

**2) NCr using dynamic programming**

#include<stdio.h>

int min(int a, int b);

int binomialCoeff(int n, int k)

{

    int C[n+1][k+1];

    int i, j;

    for (i = 0; i <= n; i++)

    {

        for (j = 0; j <= min(i, k); j++)

        {

            if (j == 0 || j == i)

                C[i][j] = 1;

            else

                C[i][j] = C[i-1][j-1] + C[i-1][j];

        }

    }

    return C[n][k];

}

int min(int a, int b)

{    return (a<b)? a: b; }

int main()

{

    int n = 5, k = 2;

    printf ("Value of C(%d, %d) is %d ", n, k, binomialCoeff(n, k) );

    return 0;

}

**3) Heap sort using C**

#include<stdio.h>

void create(int []);

void down\_adjust(int [],int);

void main()

{

    int heap[30],n,i,last,temp;

    printf("Enter no. of elements:");

    scanf("%d",&n);

    printf("\nEnter elements:");

    for(i=1;i<=n;i++)

        scanf("%d",&heap[i]);

    heap[0]=n;

    create(heap);

    while(heap[0] > 1)

    {

        last=heap[0];

        temp=heap[1];

        heap[1]=heap[last];

        heap[last]=temp;

        heap[0]--;

        down\_adjust(heap,1);

    }

    printf("\nArray after sorting:\n");

    for(i=1;i<=n;i++)

        printf("%d ",heap[i]);

}

void create(int heap[])

{

    int i,n;

    n=heap[0];

    for(i=n/2;i>=1;i--)

        down\_adjust(heap,i);

}

void down\_adjust(int heap[],int i)

{

    int j,temp,n,flag=1;

    n=heap[0];

    while(2\*i<=n && flag==1)

    {

        j=2\*i;

        if(j+1<=n && heap[j+1] > heap[j])

            j=j+1;

        if(heap[i] > heap[j])

            flag=0;

        else

        {

            temp=heap[i];

            heap[i]=heap[j];

            heap[j]=temp;

            i=j;

        }

    }

}

**Example: Factorial of a Number Using Recursion**

#include <stdio.h>

long int multiplyNumbers(int n);

int main()

{

int n;

printf("Enter a positive integer: ");

scanf("%d", &n);

printf("Factorial of %d = %ld", n, multiplyNumbers(n));

return 0;

}

long int multiplyNumbers(int n)

{

if (n >= 1)

return n\*multiplyNumbers(n-1);

else

return 1;

}

**Program to find maximum or minimum element in array**

#include <stdio.h>

int main()

{

int arr[100];

int i, max, min, size;

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

scanf("%d", &size);

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

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

{

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

}

max = arr[0];

min = arr[0];

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

{

if(arr[i]>max)

{

max = arr[i];

}

if(arr[i]<min)

{

min = arr[i];

}

}

printf("Maximum element = %d\n", max);

printf("Minimum element = %d", min);

return 0;

}

**Bubble sort algorithm in c**

#include <stdio.h>

int main()

{

int array[100], n, c, d, swap;

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

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

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

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

{

for (d = 0 ; d < n - c - 1; d++)

{

if (array[d] > array[d+1]) /\* For decreasing order use < \*/

{

swap = array[d];

array[d] = array[d+1];

array[d+1] = swap;

}

}

}

printf("Sorted list in ascending order:\n");

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

printf("%d\n", array[c]);

return 0;

}

**C program to print all unique elements of an unsorted array**

#include<stdio.h>

#include<conio.h>

int main()

{

int array[100], size, i, j;

printf("Enter number of elements in array\n");

scanf("%d", &size);

printf("Enter %d numbers\n", size);

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

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

}

printf("Unique Elements\n");

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

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

if (array[i] == array[j])

break;

}

if (i == j){

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

}

}

return 0;

}

**// C program for implementation of selection sort**

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

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;

swap(&arr[min\_idx], &arr[i]);

}

}

void printArray(int arr[], int size)

{

int i;

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

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

printf("\n");

}

int main()

{

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

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

selectionSort(arr, n);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

**Matrix multiplication in c language**

#include <stdio.h>

int main()

{

int m, n, p, q, c, d, k, sum = 0;

int first[10][10], second[10][10], multiply[10][10];

printf("Enter the number of rows and columns of first matrix\n");

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

printf("Enter the elements of first matrix\n");

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

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

scanf("%d", &first[c][d]);

printf("Enter the number of rows and columns of second matrix\n");

scanf("%d%d", &p, &q);

if (n != p)

printf("Matrices with entered orders can't be multiplied with each other.\n");

else

{

printf("Enter the elements of second matrix\n");

for (c = 0; c < p; c++)

for (d = 0; d < q; d++)

scanf("%d", &second[c][d]);

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

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

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

sum = sum + first[c][k]\*second[k][d];

}

multiply[c][d] = sum;

sum = 0;

}

}

printf("Product of entered matrices:-\n");

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

for (d = 0; d < q; d++)

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

printf("\n");

}

}

return 0;

}

**Linear search c program**

#include <stdio.h>

int main()

{

int array[100], search, c, n;

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

scanf("%d",&n);

printf("Enter %d integer(s)\n", n);

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

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

printf("Enter the number to search\n");

scanf("%d", &search);

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

{

if (array[c] == search)

{

printf("%d is present at location %d.\n", search, c+1);

break;

}

}

if (c == n)

printf("%d is not present in array.\n", search);

return 0;

}

**C programming code for binary search**

#include <stdio.h>

int main()

{

int c, first, last, middle, n, search, array[100];

printf("Enter number of elements\n");

scanf("%d",&n);

printf("Enter %d integers\n", n);

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

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

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last) {

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search) {

printf("%d found at location %d.\n", search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found! %d is not present in the list.\n", search);

return 0; }

**GCD of Two Numbers using Recursion**

#include <stdio.h>

int hcf(int n1, int n2);

int main()

{

int n1, n2;

printf("Enter two positive integers: ");

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

printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1,n2));

return 0;

}

int hcf(int n1, int n2)

{

if (n2 != 0)

return hcf(n2, n1%n2);

else

return n1;

}

**C Program to Solve Tower-of-Hanoi Problem using Recursion**

#include <stdio.h>

void towers(int, char, char, char);

int main()

{

int num;

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

scanf("%d", &num);

printf("The sequence of moves involved in the Tower of Hanoi are :\n");

towers(num, 'A', 'C', 'B');

return 0;

}

void towers(int num, char frompeg, char topeg, char auxpeg)

{

if (num == 1)

{

printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);

return;

}

towers(num - 1, frompeg, auxpeg, topeg);

printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);

towers(num - 1, auxpeg, topeg, frompeg);

}

**C Program to Convert a Decimal Number to Binary & Count the Number of 1s**

#include <stdio.h>

void main()

{

long num, decimal\_num, remainder, base = 1, binary = 0, no\_of\_1s = 0;

printf("Enter a decimal integer \n");

scanf("%ld", &num);

decimal\_num = num;

while (num > 0)

{

remainder = num % 2;

if (remainder == 1)

{

no\_of\_1s++;

}

binary = binary + remainder \* base;

num = num / 2;

base = base \* 10;

}

printf("Input number is = %d\n", decimal\_num);

printf("Its binary equivalent is = %ld\n", binary);

printf("No.of 1's in the binary number is = %d\n", no\_of\_1s); }

**Quick Sort Program in C**

#include <stdio.h>

void quick\_sort(int[],int,int);

int partition(int[],int,int);

int main()

{

int a[50],n,i;

printf("How many elements?");

scanf("%d",&n);

printf("\nEnter array elements:");

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

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

quick\_sort(a,0,n-1);

printf("\nArray after sorting:");

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

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

return 0;

}

void quick\_sort(int a[],int l,int u)

{

int j;

if(l<u)

{

j=partition(a,l,u);

quick\_sort(a,l,j-1);

quick\_sort(a,j+1,u);

}

}

int partition(int a[],int l,int u)

{

int v,i,j,temp;

v=a[l];

i=l;

j=u+1;

do

{

do

i++;

while(a[i]<v&&i<=u);

do

j--;

while(v<a[j]);

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}while(i<j);

a[l]=a[j];

a[j]=v;

return(j);

}

**Merge sort**

#include<stdlib.h>

#include<stdio.h>

void merge(int arr[], int l, int m, int r)

{

int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

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

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

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

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

i = 0; // Initial index of first subarray

j = 0; // Initial index of second subarray

k = l; // Initial index of merged subarray

while (i < n1 && j < n2)

{

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

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void mergeSort(int arr[], int l, int r)

{

if (l < r)

{

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

mergeSort(arr, l, m);

mergeSort(arr, m+1, r);

merge(arr, l, m, r);

}

}

void printArray(int A[], int size)

{

int i;

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

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

printf("\n");

}

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; }