**SELECTION SORT:**

#include<stdio.h>

#include<conio.h>

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

int minId;

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

minId=m;

for(int l=m+1;l<n;l++){

if(arr[l]<arr[minId])

minId=l;

}

int temp=arr[minId];

arr[minId]=arr[m];

arr[m]=temp;

}

}

int main(){

int n;

clrscr();

printf("\nEnter the number of elements you want in an array:");

scanf("%d",&n);

int arr[10];

printf("\nEnter the elements of array\n");

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

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

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

}

printf("\n\nDisplaying the entered array:\n");

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

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

}

selection\_sort(arr,n);

printf("\n\nDisplaying the sorted array:\n");

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

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

}

getch();

return 0;

}

**INSERTION SORT:**

#include<stdio.h>

#include<conio.h>

int main(){

int n;

clrscr();

int arr[20];

printf("Enter the total no of elements you want in an array:");

scanf("%d",&n);

printf("\nEnter the elements\n");

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

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

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

}

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

int temp,y;

temp=arr[x];

y=x-1;

while(y>=0 && arr[y]>temp){

arr[y+1]=arr[y];

y--;

}

arr[y+1]=temp;

}

printf("\nAfter sorting:\n");

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

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

}

getch();

return 0;

}

**MERGE SORT:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

void Merge(int arr[], int lb, int mid, int ub) {

int i, j, k;

int b[20];

i = lb;

j = mid + 1;

k = lb;

while (i <= mid && j <= ub) {

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

b[k] = arr[i];

i++;

} else {

b[k] = arr[j];

j++;

}

k++;

}

while (i <= mid) {

b[k] = arr[i];

i++;

k++;

}

while (j <= ub) {

b[k] = arr[j];

j++;

k++;

}

for (k = lb; k <= ub; k++) {

arr[k] = b[k];

}

}

void Merge\_sort(int arr[], int lb, int ub) {

int mid;

if (lb < ub) {

mid = (lb + ub) / 2;

Merge\_sort(arr, lb, mid);

Merge\_sort(arr, mid + 1, ub);

Merge(arr, lb, mid, ub);

}

}

int main() {

int size;

clrscr();

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

scanf("%d", &size);

printf("\n");

int arr[20];

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

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

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

}

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

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

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

}

printf("\n");

Merge\_sort(arr, 0, size - 1);

printf("\n\nThe elements of the array after sorting:");

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

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

}

printf("\n");

getch();

return 0;

}

**MINMAX:**

#include<stdio.h>

#include<conio.h>

int main() {

int size,max,min;

clrscr();

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

scanf("%d", &size);

printf("\n");

int arr[20];

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

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

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

}

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

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

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

}

printf("\n");

min =max = arr[0];

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

if(min > arr[j]){

min = arr[j];

}

else if(max < arr[j]){

max = arr[j];

}

}

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

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

getch();

return 0;

}

**BINARY SEARCH:**

#include <stdio.h>

// Binary search function

int binarySearch(int array[], int x, int low, int high) {

while (low <= high) {

int mid = low + (high - low) / 2;

if (array[mid] == x)

return mid;

if (array[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

return -1;

}

int main() {

int n;

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

scanf("%d", &n);

int array[20];

printf("Enter %d elements in the array:\n", n);

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

printf("Element[%d]: ", i);

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

}

printf("\nEntered array:\n");

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

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

}

printf("\n");

printf("\nEnter the key value: ");

int x;

scanf("%d", &x);

int result = binarySearch(array, x, 0, n - 1);

if (result == -1)

printf("Element not found\n");

else

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

return 0;

}

**KANPSACK PROBLEM:**

#include<stdio.h>

#include<conio.h>

int main() {

float weight[50], profit[50], ratio[50], temp, capacity, Totalvalue = 0;

int n, i, j;

clrscr();

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

scanf("%d", &n);

if (n <= 0 || n > 50) {

printf("Invalid number of items\n");

return 1;

}

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

printf("Enter Weight and Profit for item[%d]: ", i);

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

}

printf("Enter the capacity of knapsack: ");

scanf("%f", &capacity);

if (capacity <= 0) {

printf("Invalid capacity\n");

return 1;

}

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

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

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

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

}

printf("Knapsack problems using Greedy Algorithm:\n");

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

if (weight[i] > capacity)

break;

else {

Totalvalue = Totalvalue + profit[i];

capacity = capacity - weight[i];

}

}

if (i < n)

Totalvalue = Totalvalue + (ratio[i] \* capacity);

printf("\nThe profit value is: %.2f\n", Totalvalue);

getch();

return 0;

}

**N-QUEEN:**

#include <stdbool.h>

#include <stdio.h>

#define N 4

void nQueens() {

int board[N][N] = {

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 },

{ 0, 0, 0, 0 }

};

bool isSafe(int board[N][N], int row, int col) {

int i, j;

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

if (board[row][i])

return false;

for (i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return false;

for (i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return false;

return true;

}

bool solveNQUtil(int board[N][N], int col) {

if (col >= N)

return true;

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

if (isSafe(board, i, col)) {

board[i][col] = 1;

if (solveNQUtil(board, col + 1))

return true;

board[i][col] = 0;

}

}

return false;

}

void printSolution(int board[N][N]) {

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

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

if (board[i][j])

printf("Q ");

else

printf(". ");

}

printf("\n");

}

}

if (solveNQUtil(board, 0) == false) {

printf("Solution does not exist");

return;

}

printSolution(board);

}

int main() {

nQueens();

return 0;

}

**STRING MATCHING:**

#include <stdio.h>

#include <conio.h>

#include <string.h>

int main() {

char text[100];

char pattern[100];

clrscr();

printf("Enter the text: ");

scanf("%s", text);

printf("Enter the pattern: ");

scanf("%s", pattern);

int textLength = strlen(text);

int patternLength = strlen(pattern);

for (int i = 0; i <= textLength - patternLength; i++) {

int j;

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

if (text[i + j] != pattern[j])

break;

}

if (j == patternLength)

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

}

getch();

return 0;

}