C Programs: Data Structures and Algorithms

## 1. Linear Search

#include <stdio.h>  
int linear\_search(int arr[], int n, int key) {  
 for (int i = 0; i < n; i++) {  
 if (arr[i] == key)  
 return i;  
 }  
 return -1;  
}

## 2. Binary Search

int binary\_search(int arr[], int n, int key) {  
 int low = 0, high = n - 1;  
 while (low <= high) {  
 int mid = (low + high) / 2;  
 if (arr[mid] == key)  
 return mid;  
 else if (arr[mid] < key)  
 low = mid + 1;  
 else  
 high = mid - 1;  
 }  
 return -1;  
}

## 3. Bubble Sort (Ascending)

void bubble\_sort(int arr[], int n) {  
 for (int i = 0; i < n - 1; i++) {  
 for (int j = 0; j < n - i - 1; j++) {  
 if (arr[j] > arr[j + 1]) {  
 int temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
 }  
 }  
 }  
}

## 4. Selection Sort (Descending)

void selection\_sort\_desc(int arr[], int n) {  
 for (int i = 0; i < n - 1; i++) {  
 int max = i;  
 for (int j = i + 1; j < n; j++) {  
 if (arr[j] > arr[max])  
 max = j;  
 }  
 int temp = arr[i];  
 arr[i] = arr[max];  
 arr[max] = temp;  
 }  
}

## 5. Insertion Sort (Ascending)

void insertion\_sort(int arr[], int n) {  
 for (int i = 1; i < n; i++) {  
 int key = arr[i], j = i - 1;  
 while (j >= 0 && arr[j] > key) {  
 arr[j + 1] = arr[j];  
 j--;  
 }  
 arr[j + 1] = key;  
 }  
}

## 6. Linear Queue Operations

#define MAX 100  
int queue[MAX], front = -1, rear = -1;  
void enqueue(int val) {  
 if (rear == MAX - 1)  
 printf("Queue Overflow\n");  
 else {  
 if (front == -1) front = 0;  
 queue[++rear] = val;  
 }  
}  
void dequeue() {  
 if (front == -1 || front > rear)  
 printf("Queue Underflow\n");  
 else  
 front++;  
}  
void display\_queue() {  
 if (front == -1 || front > rear)  
 printf("Queue is empty\n");  
 else {  
 for (int i = front; i <= rear; i++)  
 printf("%d ", queue[i]);  
 printf("\n");  
 }  
}

## 7. Stack Operations

int stack[MAX], top = -1;  
void push(int val) {  
 if (top == MAX - 1)  
 printf("Stack Overflow\n");  
 else  
 stack[++top] = val;  
}  
void pop() {  
 if (top == -1)  
 printf("Stack Underflow\n");  
 else  
 top--;  
}  
void display\_stack() {  
 for (int i = 0; i <= top; i++)  
 printf("%d ", stack[i]);  
 printf("\n");  
}

## 8. GCD using Recursion

int gcd(int a, int b) {  
 if (b == 0)  
 return a;  
 return gcd(b, a % b);  
}

## 9. Towers of Hanoi

void hanoi(int n, char from, char to, char aux) {  
 if (n == 1) {  
 printf("Move disk 1 from %c to %c\n", from, to);  
 return;  
 }  
 hanoi(n - 1, from, aux, to);  
 printf("Move disk %d from %c to %c\n", n, from, to);  
 hanoi(n - 1, aux, to, from);  
}

## 10. Sum of Array using Recursion

int sum\_array(int arr[], int n) {  
 if (n <= 0)  
 return 0;  
 return arr[n - 1] + sum\_array(arr, n - 1);  
}

## 11. Delete element in array

void delete\_element(int arr[], int \*n, int pos) {  
 for (int i = pos; i < \*n - 1; i++)  
 arr[i] = arr[i + 1];  
 (\*n)--;  
}

## 12. Insert element in array

void insert\_element(int arr[], int \*n, int pos, int val) {  
 for (int i = \*n; i > pos; i--)  
 arr[i] = arr[i - 1];  
 arr[pos] = val;  
 (\*n)++;  
}