1.Topological sort

Using dfs method

```
#include <stdio.h>
void findindegree(int n, int a[][10], int indegree[10]) {
int i, j;
for(i = 0; i < n; i++) {
indegree[i] = 0; // Initialize indegree array
}
for(i = 0; i < n; i++) {
for(j = 0; j < n; j++) {
if(a[j][i] == 1) {
indegree[i]++;
}
}
}
}

void topological(int n, int a[][10]) {
int i, u, v, top, k, t[10], indegree[10], s[10];
findindegree(n, a, indegree);
top = -1;
k = 0;
for(i = 0; i < n; i++) {
if(indegree[i] == 0) {
s[++top] = i;
}
```

```
}
while(top != -1) {
u = s[top--];
t[k++] = u;
for(v = 0; v < n; v++) {
if(a[u][v] == 1) {
indegree[v]--;
if(indegree[v] == 0) {
s[++top] = v;
}
}
}
}
printf("Topological order\n");
for(i = 0; i < k; i++) {
printf("%d\t", t[i]);
}
printf("\n");
}

int main() {
int n, b[10][10];
int i, j;
printf("Enter the number of jobs\n");
scanf("%d", &n);
printf("Enter the adjacency matrix\n");
```

```
for(i = 0; i < n; i++) {
for(j = 0; j < n; j++) {
scanf("%d", &b[i][j]);
}
}
topological(n, b);
return 0;
}
```

Output:

```
Enter the number of nodes:4
Enter the number of edges
3
Enter the edge i,j
0 1
Enter the edge i,j
1 2
Enter the edge i,j
2 3
The topological sequence
0123
```

2.GCD using recursion:

```
#include <stdio.h>

// Function to find GCD using recursion
int gcd(int a, int b) {
if (b == 0)
return a;
return gcd(b, a % b);
}

int main() {
int num1, num2;

printf("Enter two numbers: ");
scanf("%d %d", &num1, &num2);

int result = gcd(num1, num2);
printf("GCD of %d and %d is %d\n", num1, num2, result);
return 0;
}
```

Output:

Enter two numbers: 56 98

GCD of 56 and 98 is 14

3.Towers of Hanoi

```
#include <stdio.h>

// Recursive function to solve the Towers of Hanoi puzzle
void towersOfHanoi(int n, char from_rod, char to_rod, char aux_rod) {
    if (n == 1) {
        printf("Move disk 1 from rod %c to rod %c\n", from_rod, to_rod);
        return;
    }
    towersOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    printf("Move disk %d from rod %c to rod %c\n", n, from_rod, to_rod);
    towersOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

int main() {
    int n;

    printf("Enter the number of disks: ");
    scanf("%d", &n);

    printf("The sequence of moves involved in the Tower of Hanoi are:\n");
    towersOfHanoi(n, 'A', 'C', 'B');

    return 0;
}
```

Output:

Enter the number of disks: 3

The sequence of moves involved in the Tower of Hanoi are:

Move disk 1 from rod A to rod C

Move disk 2 from rod A to rod B

Move disk 1 from rod C to rod B

Move disk 3 from rod A to rod C

Move disk 1 from rod B to rod A

Move disk 2 from rod B to rod C

Move disk 1 from rod A to rod C

4.Computing median

```
#include <stdio.h>

// Function to swap two elements
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

// Lomuto partition function
int lomutoPartition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
```

```
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return i + 1;
}

// Quickselect function to find the k-th smallest element
int quickSelect(int arr[], int low, int high, int k) {
    if (low <= high) {
        int pivotIndex = lomutoPartition(arr, low, high);
        if (pivotIndex == k)
            return arr[pivotIndex];
        else if (pivotIndex > k)
            return quickSelect(arr, low, pivotIndex - 1, k);
        else
            return quickSelect(arr, pivotIndex + 1, high, k);
    }
    return -1;
}

// Function to find the median of an array
double findMedian(int arr[], int n) {
    if (n % 2 != 0) {
        return quickSelect(arr, 0, n - 1, n / 2);
```

```
    } else {
        int left = quickSelect(arr, 0, n - 1, n / 2 - 1);
        int right = quickSelect(arr, 0, n - 1, n / 2);
        return (left + right) / 2.0;
    }
}

int main() {
    int n;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    double median = findMedian(arr, n);
    printf("The median is: %.2f\n", median);
    return 0;
}
```

Output:

Enter the number of elements: 5

Enter the elements of the array:

12 3 5 7 19

The median is: 7.00