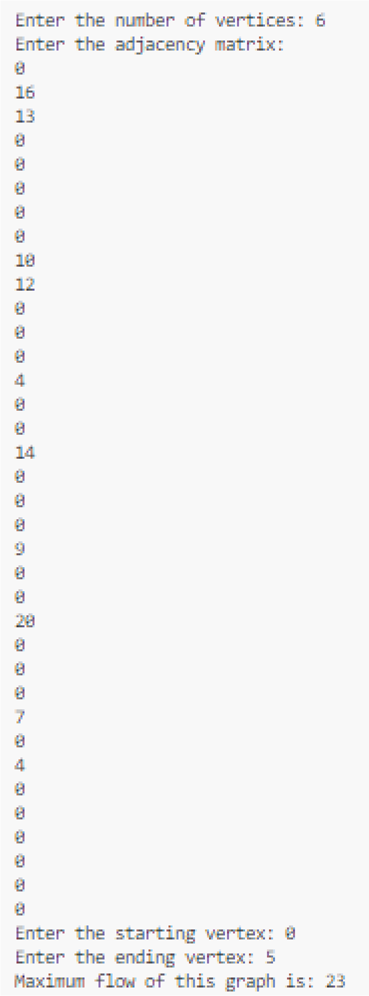
FORD FULKERSON

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>
#define MAX_SIZE 100

char color[MAX_SIZE];
int parent[MAX_SIZE];

typedef struct
{
   int *arr;
   int size;
   int rear, front;
} queue;

void init(queue *q, int n)
{
   q->rear = q->front = -1;
   q->size = n;
   q->arr = (int *)malloc(sizeof(int) * q->size);
}
int isfull(queue *q)
{
   return ((q->rear == q->size - 1 && q->front == 0) || (q->front == q->rear + 1));
}
int isempty(queue *q)
{
    return (q->rear == -1);
}
void enqueue(queue *q, int c)
{
    if (!isfull(q))
    {
        if (q->front == -1)
            q->front = 0;
        if (q->rear == q->size - 1)
            q->rear = 0;
        else
            ++q->rear;
        q->arr[q->rear] = c;
    }
}
int dequeue(queue *q)
{
    int i;
    if (isempty(q))
        i = -999;
    else
    {
        i = q->arr[q->front];
        if (q->rear == q->front)
            q->rear = q->front = -1;
        else if (q->front == q->size - 1)
            q->front = 0;
        else
            q->front = q->front + 1;
    }
    return i;
}

bool bfs(int vertices, int adj[][MAX_SIZE], int s, int d, int parent[])
{
    queue q;
    init(&q, MAX_SIZE);
    int u, v;

    for (int i = 0; i < vertices; i++)
    {
        color[i] = 'W';
        parent[i] = -1;
    }
    color[s] = 'G';
    enqueue(&q, s);
    while (!isempty(&q))
    {
        u = dequeue(&q);
        for (v = 0; v < vertices; ++v)
        {
            if (adj[u][v] > 0 && color[v] == 'W')
            {
                parent[v] = u;
                color[v] = 'G';
                if (v == d)
                {
                    return true;
                }
                else
                {
                    enqueue(&q, v);
                }
            }
        }
        color[u] = 'B';
    }
    return false;
}

int Ford_Fulkerson(int vertices, int adj[][MAX_SIZE], int s, int d)
{
    int rgraph[MAX_SIZE][MAX_SIZE];
    int u, v, maxflow, minflow;
    for (u = 0; u < vertices; ++u)
    {
        for (v = 0; v < vertices; ++v)
        {
            rgraph[u][v] = adj[u][v];
        }
    }
    maxflow = 0;
    while (bfs(vertices, rgraph, s, d, parent))
    {
        minflow = 32000;
        for (v = d; v != s; v = parent[v])
        {
            u = parent[v];
            if (rgraph[u][v] < minflow)
            {
                minflow = rgraph[u][v];
            }
        }
        for (v = d; v != s; v = parent[v])
        {
            u = parent[v];
            rgraph[u][v] = rgraph[u][v] - minflow;
            rgraph[v][u] = rgraph[v][u] + minflow;
        }
        maxflow = maxflow + minflow;
    }
    return maxflow;
}

int main()
{
    int vertices, edges, start, end;
    printf("Enter the number of vertices: ");
    scanf("%d", &vertices);
    int adj[MAX_SIZE][MAX_SIZE];
    printf("Enter the adjacency matrix:\n");
    for (int i = 0; i < vertices; i++)
    {
        for (int j = 0; j < vertices; j++)
        {
            scanf("%d", &adj[i][j]);
        }
    }
    printf("Enter the starting vertex: ");
    scanf("%d", &start);
    printf("Enter the ending vertex: ");
    scanf("%d", &end);
    int max_count = Ford_Fulkerson(vertices, adj, start, end);
    printf("Maximum flow of this graph is: %d", max_count);
    return 0;
}
```

OUTPUT



```
Enter the number of vertices: 6
Enter the adjacency matrix:
0
16
13
0
0
0
0
0
10
12
0
0
0
4
0
0
14
0
0
0
9
0
0
20
0
0
0
7
0
4
0
0
0
0
0
0
Enter the starting vertex: 0
Enter the ending vertex: 5
Maximum flow of this graph is: 23
```