Distance vector:

```
#include <stdio.h>

#define INF 999

#define MAX 10

int main() {

    int n, i, j, k, updated;

    int cost[MAX][MAX], dist[MAX][MAX];

    printf("Enter number of nodes: ");

    scanf("%d", &n);

    printf("Enter adjacency matrix (999 for INF):\n");

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

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

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

    // Initialize distance table

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

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

            dist[i][j] = cost[i][j];

    // Distance Vector Algorithm

    do {

        updated = 0;

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

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

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

                    if (dist[i][j] > dist[i][k] + dist[k][j]) {

                        dist[i][j] = dist[i][k] + dist[k][j];

                        updated = 1;

                    }

    } while (updated);

    // Print final distance table

    printf("\nFinal Distance Table:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            printf("%d ", dist[i][j]);
        printf("\n");
    }
    return 0;
}
```

Dijkstras Algorithm:

```
#include <stdio.h>
#define INF 9999
#define MAX 10
void dijkstra(int g[MAX][MAX], int n, int src) {
    int dist[MAX], visited[MAX];
    // Initialize
    for (int i = 0; i < n; i++) {
        dist[i] = INF;
        visited[i] = 0;
    }
    dist[src] = 0;
    // Main algorithm
    for (int c = 0; c < n - 1; c++) {
        int u = -1;
        // Pick minimum distance unvisited vertex
        for (int i = 0; i < n; i++)
            if (!visited[i] && (u == -1 || dist[i] < dist[u]))
                u = i;
        visited[u] = 1;
        // Relax edges
        for (int v = 0; v < n; v++)
            if (g[u][v] && dist[u] + g[u][v] < dist[v])
```

```
                dist[v] = dist[u] + g[u][v];
    }
    printf("Vertex  Distance\n");
    for (int i = 0; i < n; i++)
        printf("%d\t%d\n", i, dist[i]);
}
int main() {
    int n, g[MAX][MAX], src;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &g[i][j]);
    printf("Enter source: ");
    scanf("%d", &src);
    dijkstra(g, n, src);
    return 0;
}
```

<u>CRC-12:</u>

```
#include <stdio.h>
int main() {
    int data[100], recv[200], crc[12]={0};
    int gen[13]={1,1,0,0,0,0,0,0,0,0,0,1,1};
    int n,i,j,error=0;
    printf("Enter number of data bits: ");
    scanf("%d",&n);
    printf("Enter data bits: ");
    for(i=0;i<n;i++)
        scanf("%d",&data[i]);
```

```
for(i=0;i<n;i++) {
    int bit = crc[0] ^ data[i];
    for(j=0;j<11;j++)
        crc[j] = crc[j+1] ^ (bit & gen[j+1]);
    crc[11] = bit & gen[12];
}
printf("CRC bits: ");
for(i=0;i<12;i++)
    printf("%d",crc[i]);
printf("\nTransmitted codeword: ");
for(i=0;i<n;i++)
    printf("%d",data[i]);
for(i=0;i<12;i++)
    printf("%d",crc[i]);
printf("\nEnter received codeword: ");
for(i=0;i<n+12;i++)
    scanf("%d",&recv[i]);
for(i=0;i<12;i++)
    crc[i]=0;
for(i=0;i<n+12;i++) {
    int bit = crc[0] ^ recv[i];
    for(j=0;j<11;j++)
        crc[j] = crc[j+1] ^ (bit & gen[j+1]);
    crc[11] = bit & gen[12];
}
for(i=0;i<12;i++)
    if(crc[i]!=0) error=1;
if(error)
    printf("Error detected\n");
else
```

```
        printf("No error\n");
    return 0;
}
```

CRC-16:

```
#include <stdio.h>
int main() {
    int data[100], recv[120], crc[16], poly[17] = {1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1};
    int n, i, j, bit, error = 0;
    printf("Enter number of data bits: ");
    scanf("%d", &n);
    printf("Enter data bits: ");
    for(i = 0; i < n; i++) scanf("%d", &data[i]);
    for(i = 0; i < 16; i++) crc[i] = 0;
    for(i = 0; i < n; i++) {
        bit = data[i] ^ crc[0];
        for(j = 0; j < 15; j++)
            crc[j] = crc[j+1] ^ (bit & poly[j+1]);
        crc[15] = bit & poly[16];
    }
    printf("\nCRC-16: ");
    for(i = 0; i < 16; i++) printf("%d", crc[i]);
    printf("\nTransmitted codeword: ");
    for(i = 0; i < n; i++) printf("%d", data[i]);
    for(i = 0; i < 16; i++) printf("%d", crc[i]);
    printf("\n\nEnter received codeword: ");
    for(i = 0; i < n + 16; i++) scanf("%d", &recv[i]);
    for(i = 0; i < 16; i++) crc[i] = 0;
    for(i = 0; i < n + 16; i++) {
        bit = recv[i] ^ crc[0];
        for(j = 0; j < 15; j++)
```

```
            crc[j] = crc[j+1] ^ (bit & poly[j+1]);
        crc[15] = bit & poly[16];
    }
    for(i = 0; i < 16; i++)
        if(crc[i] != 0) error = 1;
    if(error)
        printf("\nError detected in received data.\n");
    else
        printf("\nNo error. Data received correctly.\n");
    return 0;
}
```

Hamming code:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
    int code[10], data[10];
    int m, r = 0, i, j = 0, k = 0, position, parity;
    int error = 0;
    printf("Enter no of data bits: ");
    scanf("%d", &m);
    printf("Enter data bits: ");
    for (i = 1; i <= m; i++)
        scanf("%d", &data[i]);
    while ((int)pow(2, r) < (m + r + 1))
        r++;
    int n = m + r;
    for (i = 1; i <= n; i++) {
        if (i == (int)pow(2, k)) {
            code[i] = 0;
            k++;
        } else {
            code[i] = data[j + 1];
            j++;
        }
    }
    for (i = 0; i < r; i++) {
        position = (int)pow(2, i);
        parity = 0;
        for (j = position; j <= n; j++) {
            if ((j >> i) & 1)
                parity ^= code[j];
        }
        code[position] = parity;
    }
    printf("Hamming code to send: ");
    for (i = 1; i <= n; i++)
        printf("%d ", code[i]);
    printf("\nEnter received code: ");
    for (i = 1; i <= n; i++)
        scanf("%d", &code[i]);
    for (i = 0; i < r; i++) {
        position = (int)pow(2, i);
        parity = 0;
        for (j = position; j <= n; j++) {
            if ((j >> i) & 1)
                parity ^= code[j];
        }
        if (parity)
            error += position;
    }
    if (error == 0)
        printf("No error occurred\n");
    else
        printf("Error occurred at position: %d\n", error);
    return 0;
}
```