1a.Character stuffing

```
#include <stdio.h>
#include <string.h>
void main() {
    int i = 0, j = 0, n;
    char a[50], b[100];
    printf("Enter string: ");
    scanf("%s", a);
    n = strlen(a);
    b[0] = 'd';
    b[1] = 'l';
    b[2] = 'e';
    b[3] = 's';
    b[4] = 't';
    b[5] = 'x';
    j = 6;
    while (i < n) {
        if (a[i] == 'd' && a[i + 1] == 'l' && a[i + 2] == 'e') {
            b[j++] = 'd';
            b[j++] = 'l';
            b[j++] = 'e';
            b[j++] = 'd';
            b[j++] = 'l';
            b[j++] = 'e';
            i += 3;
        } else {
            b[j++] = a[i++];
        }
```

```
    }
    b[j++] = 'd';
    b[j++] = '1';
    b[j++] = 'e';
    b[j++] = 'e';
    b[j++] = 't';
    b[j++] = 'x';
    b[j] = '\0';
    printf("\nFrame after stuffing:\n");
    printf("%s\n", b);
}
```

1b.Bitstuffing

```
#include <stdio.h>
#include <string.h>
void main() {
    char input[100], stuffed[200];
    int i, j = 0, count = 0;
    printf("Enter binary data (0s and 1s): ");
    scanf("%s", input);
    for (i = 0; input[i] != '\0'; i++) {
        stuffed[j++] = input[i];
        if (input[i] == '1') {
            count++;
            if (count == 5) {
                stuffed[j++] = '0';
                count = 0;
            }
        } else {
```

```
            count = 0;
        }
    }
    stuffed[j] = '\0';
    printf("Stuffed Data: %s\n", stuffed);
}
```

2.Checksum

```
#include <stdio.h>
void main() {
    int data[100], length, i, sum = 0, checksum;
    printf("Enter length of data: ");
    scanf("%d", &length);
    printf("Enter %d data integers:\n", length);
    for (i = 0; i < length; i++) {
        scanf("%d", &data[i]);
        sum += data[i];
    }
    checksum = ~sum;
    printf("Checksum (sender side): %d\n", checksum);
    int receivedSum = 0;
    for (i = 0; i < length; i++) {
        receivedSum += data[i];
    }
    receivedSum += checksum;
    if (receivedSum == -1)
        printf("No error: Data received correctly.\n");
    else
        printf("Error detected in received data.\n");
```

```
}
```

3.Hamming code

```
#include <stdio.h>
#include <math.h>
int main() {
    int data[10], code[20];
    int m, r = 0, i, j, k = 0, position, parity, error = 0;
    printf("Enter the number of data bits: ");
    scanf("%d", &m);
    printf("Enter the data bits (space-separated): ");
    for (i = 0; i < m; i++)
        scanf("%d", &data[i]);
    while ((int)pow(2, r) < (m + r + 1))
        r++;
    int n = m + r;
    j = 0;
    for (i = 1; i <= n; i++) {
        if (i == (int)pow(2, k)) {
            code[i] = 0;
            k++;
        } else {
            code[i] = data[j];
            j++;
        }
    }
    for (i = 0; i < r; i++) {
        position = (int)pow(2, i);
        parity = 0;
```

```
    for (j = position; j <= n; j++) {
        if (((j >> i) & 1) == 1) {
            parity ^= code[j];
        }
    }
    code[position] = parity;
}
printf("\nHamming Code (to send): ");
for (i = n; i >= 1; i--)
    printf("%d", code[i]);
printf("\n");
printf("\nEnter the received code bits: ");
for (i = n; i >= 1; 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) == 1)
            parity ^= code[j];
    }
    if (parity != 0)
        error += position;
}
if (error == 0)
    printf("No error detected!\n");
else {
    printf("Error detected at bit position: %d\n", error);
    code[error] = (code[error] == 0) ? 1 : 0;
```

```
        printf("Corrected code: ");
        for (i = n; i >= 1; i--)
            printf("%d", code[i]);
        printf("\n");
    }
    return 0;
}
```

4.CRC-12

```
#include <stdio.h>
int main() {
    int data[50], crc[12], n, i, j, error = 0;
    int generator[13] = {1,1,0,0,0,0,0,0,0,0,0,1,1};
    printf("Enter number of data bits: ");
    scanf("%d", &n);
    printf("Enter data bits (0 or 1): ");
    for(i = 0; i < n; i++)
        scanf("%d", &data[i])
    for(i = 0; i < 12; i++)
        crc[i] = 0;
    for(i = 0; i < n; i++) {
        int bit = data[i] ^ crc[0];
        for(j = 0; j < 11; j++)
            crc[j] = crc[j+1] ^ (bit & generator[j+1]);
        crc[11] = bit & generator[12];
    }
    printf("\nCRC 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("\n\nEnter received codeword (data + CRC): ");
    int received[62];
    for(i = 0; i < n + 12; i++)
        scanf("%d", &received[i]);

    for(i = 0; i < n; i++) {
        int bit = received[i] ^ crc[0];
        for(j = 0; j < 11; j++)
            crc[j] = crc[j+1] ^ (bit & generator[j+1]);
        crc[11] = bit & generator[12];
    }
    for(i = 0; i < 12; i++) {
        if(crc[i] != 0)
            error = 1;
    }
    if(error)
        printf("Error detected in received data!\n");
    else
        printf("No error detected. Data received correctly.\n");
    return 0;
}
```

5.GoBackN

```
#include <stdio.h>
```

```
void main() {
    int total_frames, window_size;
    int sent = 0, lost_frame, ack;
    int i;
    printf("Enter total number of frames to send: ");
    scanf("%d", &total_frames);
    printf("Enter window size: ");
    scanf("%d", &window_size);
    printf("\n--- Go-Back-N Sliding Window Protocol Simulation ---\n\n");
    while (sent < total_frames) {
        printf("Sender: Sending frames ");
        for (i = sent; i < sent + window_size && i < total_frames; i++)
            printf("%d ", i);
        printf("\n");
        printf("Enter the frame number to be lost (or -1 if none lost): ");
        scanf("%d", &lost_frame);
        if (lost_frame >= sent && lost_frame < sent + window_size && lost_frame < total_frames) {
            printf("Receiver: Frame %d lost! Go back and resend from %d\n\n", lost_frame, lost_frame);
            sent = lost_frame;
        } else {
            ack = sent + window_size;
            if (ack > total_frames)
                ack = total_frames;
            printf("Receiver: Acknowledged up to frame %d\n\n", ack - 1);
            sent = ack;
        }
    }
    printf("All frames sent successfully using Go-Back-N!\n") }
```

6.Selective Repeat

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 50
int main() {
    int totalFrames, windowSize;
    int frameStatus[MAX];
    int i, choice, resend;
    printf("\n=== Selective Repeat ARQ Simulation ===\n");
    printf("Enter total number of frames to send: ");
    scanf("%d", &totalFrames);
    printf("Enter window size: ");
    scanf("%d", &windowSize);
    for(i = 0; i < totalFrames; i++)
        frameStatus[i] = 0;
    int sent = 0;
    while(sent < totalFrames) {
        printf("\n--- Sending Window of Frames ---\n");
        for(i = sent; i < sent + windowSize && i < totalFrames; i++) {
            if(frameStatus[i] == 0) {
                printf("Frame %d sent.\n", i);
            }
        }
        for(i = sent; i < sent + windowSize && i < totalFrames; i++) {
            if(frameStatus[i] == 0) {
                printf("Did frame %d get ACK? (1 = Yes, 0 = Lost): ", i);
                scanf("%d", &choice);
                if(choice == 1) {
```

```
            printf("ACK for frame %d received.\n", i);
            frameStatus[i] = 1;
        } else {
            printf("Frame %d lost / not acknowledged.\n", i);
        }
      }
    }
    printf("\nDo you want to resend any lost frame? (Enter frame number, -1 to stop): ");
    scanf("%d", &resend);
    while(resend != -1) {
      if(resend >= sent && resend < totalFrames && frameStatus[resend] == 0) {
        printf("Resending frame %d...\n", resend);
        printf("ACK for frame %d received now.\n", resend);
        frameStatus[resend] = 1;
      } else {
        printf("Invalid frame number or already ACKed.\n");
      }
      printf("Enter next frame to resend (-1 to stop): ");
      scanf("%d", &resend);
    }
    while(sent < totalFrames && frameStatus[sent] == 1)
      sent++;
  }
  printf("\nAll frames sent and acknowledged successfully!\n");
  return 0;
}
```

7.Stop and wait

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <time.h>
void sender(int totalFrames);
void receiver(int frame);
int main() {
    int totalFrames;
    printf("=== Stop-and-Wait Protocol Simulation ===\n");
    printf("Enter total number of frames to send: ");
    scanf("%d", &totalFrames);
    sender(totalFrames);
    return 0;
}
void sender(int totalFrames) {
    int i = 1, choice;
    srand(time(NULL));
    while (i <= totalFrames) {
        printf("\nSender: Sending Frame %d", i);
        printf("\nReceiver: Do you want to ACK Frame %d? (1-Yes, 0-No): ", i);
        scanf("%d", &choice);
        if (choice == 1) {
            receiver(i);
            printf("Sender: ACK %d received\n", i);
            i++;
        } else {
            printf("Sender: ACK %d lost. Retransmitting...\n", i);
        }
    }
    printf("\nAll %d frames sent successfully!\n", totalFrames);
}
```

```
void receiver(int frame) {
    printf("Receiver: Frame %d received. Sending ACK %d...\n", frame, frame);
}
```

8.Leaky Bucket

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int bucketSize, outputRate, n, incoming;
    int stored = 0;
    printf("=== Leaky Bucket Algorithm Simulation ===\n");
    printf("Enter bucket capacity (in packets): ");
    scanf("%d", &bucketSize);
    printf("Enter output rate (packets per second): ");
    scanf("%d", &outputRate);
    printf("Enter number of incoming packets (time slots): ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        printf("\nTime %d: Enter number of incoming packets: ", i);
        scanf("%d", &incoming);
        printf("Incoming packets: %d\n", incoming);
        stored += incoming;
        if (stored > bucketSize) {
            int dropped = stored - bucketSize;
            stored = bucketSize;
            printf("Bucket overflow! Dropped packets: %d\n", dropped);
        }
        int transmitted;
        if (stored >= outputRate) {
```

```
            transmitted = outputRate;
            stored -= outputRate;
        } else {
            transmitted = stored;
            stored = 0;
        }
        printf("Transmitted: %d | Packets left in bucket: %d\n", transmitted, stored);
    }
    while (stored > 0) {
        int transmitted;
        if (stored >= outputRate) {
            transmitted = outputRate;
            stored -= outputRate;
        } else {
            transmitted = stored;
            stored = 0;
        }
        printf("\nTransmitted: %d | Packets left in bucket: %d\n", transmitted, stored);
    }
    printf("\nAll packets transmitted successfully!\n");
    return 0;
}
```

9.Dijkstr's Algorithm

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
int minDistance(int dist[], bool visited[], int V) {
    int min = INT_MAX, min_index = -1;
```

```
    for (int i = 0; i < V; i++)
        if (!visited[i] && dist[i] <= min) {
            min = dist[i];
            min_index = i;
        }
    return min_index;
}
void printPath(int parent[], int j) {
    if (parent[j] == -1) {
        printf("%d", j);
        return;
    }
    printPath(parent, parent[j]);
    printf(" -> %d", j);
}
void dijkstra(int graph[20][20], int V, int src) {
    int dist[20];
    bool visited[20];
    int parent[20];
    for (int i = 0; i < V; i++) {
        dist[i] = INT_MAX;
        visited[i] = false;
        parent[i] = -1;
    }

    dist[src] = 0;
    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, visited, V);
        visited[u] = true;
```

```
        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u][v] != 0 && dist[u] != INT_MAX &&
                dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
                parent[v] = u;
            }
        }
    }
    printf("\nVertex\tDistance\tPath\n");
    for (int i = 0; i < V; i++) {
        if (dist[i] == INT_MAX)
            printf("%d\tINF\t\tNo path\n", i);
        else {
            printf("%d\t%d\t\t", i, dist[i]);
            printPath(parent, i);
            printf("\n");
        }
    }
}
int main() {
    int V, src;
    int graph[20][20];
    printf("Enter number of vertices (max 20): ");
    scanf("%d", &V);
    printf("\nEnter the adjacency matrix (use 0 for no edge): \n");
    for (int i = 0; i < V; i++)
        for (int j = 0; j < V; j++)
            scanf("%d", &graph[i][j]);
    printf("\nEnter the source vertex (0 to %d): ", V - 1);
```

```
    scanf("%d", &src);
    dijkstra(graph, V, src);
    return 0;
}
```