

C PROGRAMS – CN

10. SLIDING WINDOW PROTOCOL

Program :

```
#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define WINDOW_SIZE 4

#define MAX_PACKET_LENGTH 100

// Function to simulate sending a packet
void sendPacket(char packet[]) {
    printf("Sending packet: %s\n", packet);
}

// Function to simulate receiving an acknowledgment
void receiveAcknowledgment(char packet[]) {
    printf("Received acknowledgment: %s\n", packet);
}

int main() {
    char senderBuffer[WINDOW_SIZE][MAX_PACKET_LENGTH];
    int base = 0;
    int nextSeqNum = 0;
    bool isWindowFull = false;

    int totalPackets;
    printf("Enter the total number of packets to send: ");
    scanf("%d", &totalPackets);
```

```
getchar(); // To consume the newline character after the integer input
```

```
printf("Sliding Window Protocol Simulation\n");
```

```
while (base < totalPackets) {
```

```
    // Send packets if the window is not full
```

```
    while (nextSeqNum < base + WINDOW_SIZE && nextSeqNum < totalPackets) {
```

```
        char packet[MAX_PACKET_LENGTH];
```

```
        printf("Enter packet %d: ", nextSeqNum);
```

```
        fgets(packet, sizeof(packet), stdin);
```

```
        packet[strcspn(packet, "\n")] = '\0'; // Remove trailing newline character
```

```
        sendPacket(packet);
```

```
        strcpy(senderBuffer[nextSeqNum % WINDOW_SIZE], packet);
```

```
        nextSeqNum++;
```

```
    }
```

```
    // Simulate receiving an acknowledgment automatically
```

```
    if (base < totalPackets) {
```

```
        receiveAcknowledgment(senderBuffer[base % WINDOW_SIZE]);
```

```
        base++;
```

```
        printf("Acknowledgment accepted.\n");
```

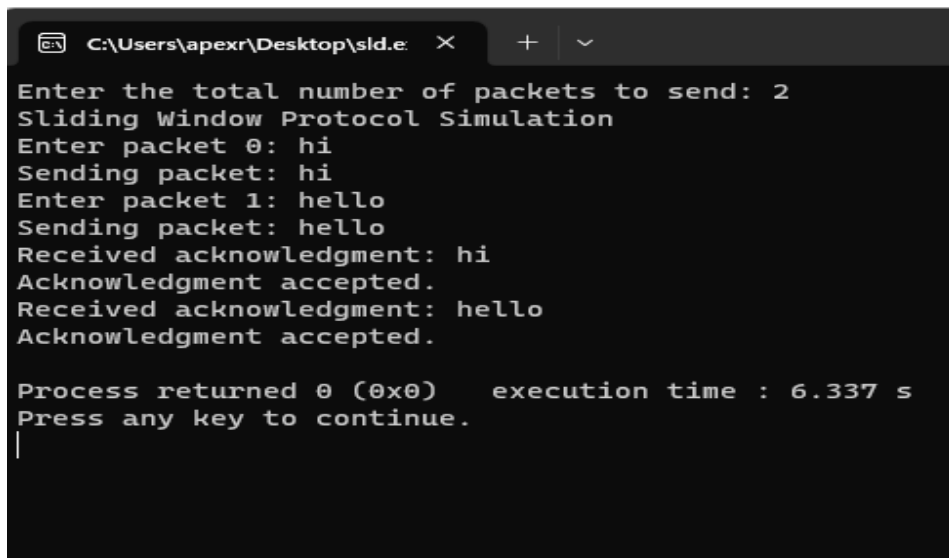
```
    }
```

```
}
```

```
return 0;
```

```
}
```

output : (input should be given by user)



```
C:\Users\apexr\Desktop\sld.e  X  +  v
Enter the total number of packets to send: 2
Sliding Window Protocol Simulation
Enter packet 0: hi
Sending packet: hi
Enter packet 1: hello
Sending packet: hello
Received acknowledgment: hi
Acknowledgment accepted.
Received acknowledgment: hello
Acknowledgment accepted.

Process returned 0 (0x0)   execution time : 6.337 s
Press any key to continue.
|
```

9. SIMULATION OF ERROR CORRECTION CODE – CRC

Program:

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

```
#include<string.h>
```

```
// Function to perform CRC error detection
```

```
void crcErrorDetection(char data[], char divisor[]) {
```

```
    int dataLength = strlen(data);
```

```
    int divisorLength = strlen(divisor);
```

```
    // Append zeros to the data
```

```
    for (int i = 0; i < divisorLength - 1; i++) {
```

```
        data[dataLength + i] = '0';
```

```
    }
```

```
    data[dataLength + divisorLength - 1] = '\0';
```

```
    // Perform division
```

```
    for (int i = 0; i < dataLength; i++) {
```

```
        if (data[i] == '1') {
```

```
            // XOR the divisor with the current data
```

```

        for (int j = 0; j < divisorLength; j++) {
            data[i + j] = (data[i + j] == divisor[j]) ? '0' : '1';
        }
    }
}

```

```

// Check if an error exists

```

```

int error = 0;
for (int i = dataLength; i < dataLength + divisorLength - 1; i++) {
    if (data[i] == '1') {
        error = 1;
        break;
    }
}

```

```

// Print the results

```

```

printf("Data: %s\n", data);
printf("Divisor: %s\n", divisor);
if (error) {
    printf("Error detected: An error exists in the data.\n");
} else {
    printf("No error detected: The data is error-free.\n");
}
}

```

```

int main() {
    char data[100];
    char divisor[100];

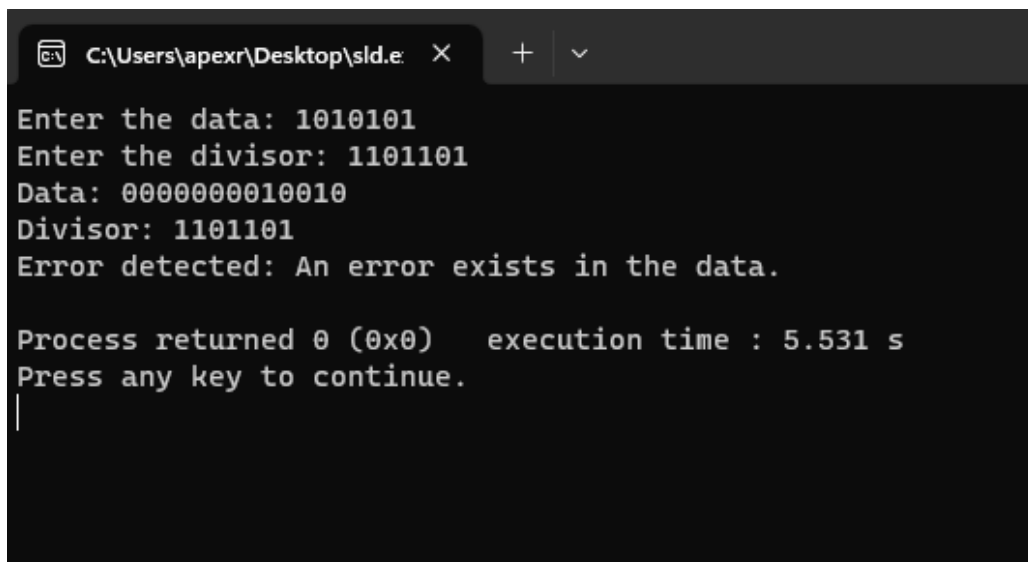
    printf("Enter the data: ");
    scanf("%s", data);
}

```

```
printf("Enter the divisor: ");  
scanf("%s", divisor);  
  
crcErrorDetection(data, divisor);  
  
return 0;  
}
```

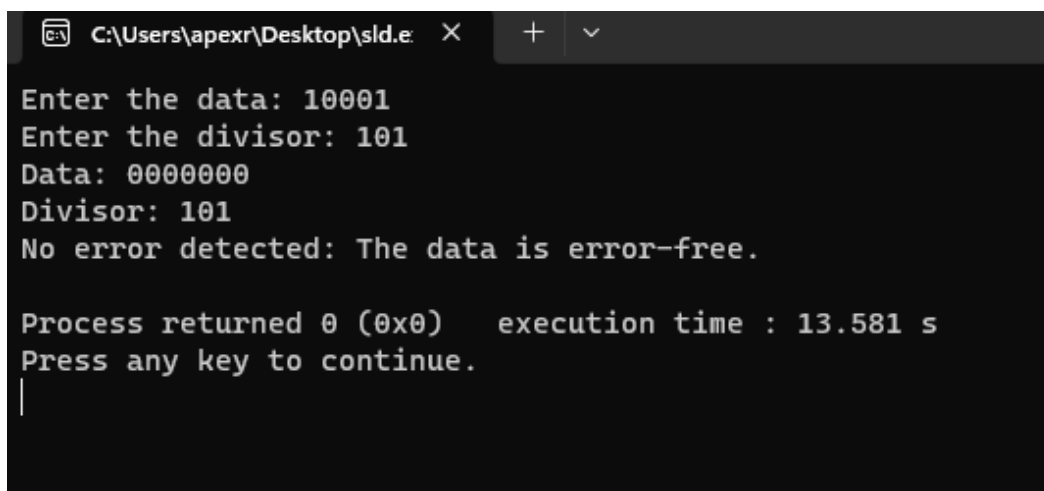
OUTPUT :

With error :



```
C:\Users\apexr\Desktop\sld.e X + v  
Enter the data: 1010101  
Enter the divisor: 1101101  
Data: 0000000010010  
Divisor: 1101101  
Error detected: An error exists in the data.  
  
Process returned 0 (0x0)   execution time : 5.531 s  
Press any key to continue.  
|
```

Without error :



```
C:\Users\apexr\Desktop\sld.e X + v  
Enter the data: 10001  
Enter the divisor: 101  
Data: 00000000  
Divisor: 101  
No error detected: The data is error-free.  
  
Process returned 0 (0x0)   execution time : 13.581 s  
Press any key to continue.  
|
```

7.DISTANCE VECTOR ROUTING

Program :

```
#include <stdio.h>

#include <stdbool.h>

#define INFINITY 9999

#define MAX_NODES 10

int distanceMatrix[MAX_NODES][MAX_NODES];
int nextHop[MAX_NODES][MAX_NODES];
int numNodes;

void initialize(int num) {
    numNodes = num;
    int i, j;

    for (i = 0; i < numNodes; i++) {
        for (j = 0; j < numNodes; j++) {
            if (i == j) {
                distanceMatrix[i][j] = 0;
                nextHop[i][j] = i;
            } else {
                distanceMatrix[i][j] = INFINITY;
                nextHop[i][j] = -1;
            }
        }
    }
}
```

```

void addLink(int source, int destination, int cost) {
    distanceMatrix[source][destination] = cost;
    distanceMatrix[destination][source] = cost;
    nextHop[source][destination] = destination;
    nextHop[destination][source] = source;
}

```

```

void distanceVectorRouting() {
    int i, j, k;

    for (k = 0; k < numNodes; k++) {
        for (i = 0; i < numNodes; i++) {
            for (j = 0; j < numNodes; j++) {
                if (distanceMatrix[i][j] > distanceMatrix[i][k] + distanceMatrix[k][j]) {
                    distanceMatrix[i][j] = distanceMatrix[i][k] + distanceMatrix[k][j];
                    nextHop[i][j] = nextHop[i][k];
                }
            }
        }
    }
}

```

```

void printRoutingTable() {
    printf("Routing Table:\n");
    printf("-----\n");
    printf("Source\tDestination\tCost\tNext Hop\n");
    printf("-----\n");

    for (int i = 0; i < numNodes; i++) {
        for (int j = 0; j < numNodes; j++) {
            printf("%d\t%d\t\t%d\t%d\n", i, j, distanceMatrix[i][j], nextHop[i][j]);
        }
    }
}

```

```

    }
}
printf("-----\n");
}

int main() {
    int num;

    printf("Enter the number of nodes: ");
    scanf("%d", &num);

    initialize(num);

    int numLinks;
    printf("Enter the number of links: ");
    scanf("%d", &numLinks);

    printf("Enter the links (source, destination, cost):\n");
    for (int i = 0; i < numLinks; i++) {
        int source, destination, cost;
        scanf("%d %d %d", &source, &destination, &cost);
        addLink(source, destination, cost);
    }

    distanceVectorRouting();
    printRoutingTable();

    return 0;
}

```


OUTPUT:

```
C:\Users\apexr\Desktop\8.exe X + v
Enter the number of nodes: 4
Enter the number of links: 5
Enter the links (source, destination, cost):
0 1 2
0 2 4
1 2 1
1 3 7
2 3 3
Routing Table:
-----
Source Destination Cost Next Hop
-----
0 0 0 0
0 1 2 1
0 2 3 1
0 3 6 1
1 0 2 0
1 1 0 1
1 2 1 2
1 3 4 2
2 0 3 1
2 1 1 1
2 2 0 2
2 3 3 3
3 0 6 2
3 1 4 2
3 2 3 2
3 3 0 3
-----
```