NAME OF THE EXPERIMENT: 2.a) Simplex protocol (stop and wait protocol)

OBJECTIVE: Implement the data link layer framing method.

RESOURCE: Code blocks

PROGRAM LOGIC: Stop and wait protocol is a simple and reliable protocol for flow control. Stop and wait protocol is a data link layer protocol. In this protocol, the sender will not send the next packet to the receiver until the acknowledgment of the previous packet is received.

Program:

```c
#include <stdio.h>
#include <stdlib.h>
void main()
{
    int i,j,n,x,x1=10,x2;
    n=10;
    i=1;j=1;
    printf("no .of frames is %d",n);
    getch();
    while(n>0)
    {
     printf("\n sending frames is %d",i);
     x=rand()%10;
     if(x%10==0)
      {
        for(x2=1;x2<2;x2++)
         {
            printf("\n waiting for %d seonds in \n ",x2);
            sleep(x2);
         }
        printf("\n sending frames is %d ",i);
        x=rand()%10;
      }
     printf("\n ack for frame is %d \n ",j);
     n=n-1;
     i++;
     j++;
    }
    printf("\n end of stop and wait protocol \n");
    getch();
}
```

Output:

```
C:\Users\GRIET\Desktop\CodeBlocks\count\bin\Debug\count.exe    -  □  ×
no .of frames is 10
sending frames is 1
ack for frame is 1

sending frames is 2
ack for frame is 2

sending frames is 3
ack for frame is 3

sending frames is 4
waiting for 1 seonds in

sending frames is 4
ack for frame is 4

sending frames is 5
ack for frame is 5

sending frames is 6
ack for frame is 6

sending frames is 7
ack for frame is 7
```

NAME OF THE EXPERIMENT: 2.b) Sliding Window protocol(Go Back N)

OBJECTIVE: Implement the data link layer framing method.
RESOURCE: Code blocks

PROGRAM LOGIC: Go-Back-N is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.

The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again.

Program:

```
#include <stdio.h>

int main() {
int total_frames, windowsize, sent = 0, ack = 0, i;
int frame_start = 0; // Start of the window

printf("Enter the total number of frames to be sent: ");
scanf("%d", &total_frames);

printf("Enter the window size: ");
scanf("%d", &windowsize);

if (windowsize <= 0 || total_frames <= 0 || windowsize > total_frames) {
printf("Invalid window size or number of frames.\n");
return 1;
}

while (sent < total_frames) {
// Transmit frames within the window
for (i = 0; i < windowsize && (frame_start + i) < total_frames; i++) {
printf("Frame %d has been transmitted.\n", frame_start + i);
}
```

```
// Prompt for the last acknowledgment received
printf("\nPlease enter the last Acknowledgement received (0 to %d): ", frame_start + i - 1);
scanf("%d", &ack);

if (ack < frame_start || ack > frame_start + i - 1) {
printf("Invalid acknowledgment. Please enter a valid acknowledgment within the transmitted window.\n");
} else if (ack >= total_frames) {
printf("Acknowledgment %d is beyond the total number of frames. Please enter a valid acknowledgment.\n", ack);
} else {
printf("Acknowledgment %d received.\n", ack);

// Move the window to start after the last acknowledged frame
frame_start = ack + 1;

// Update the `sent` variable to ensure frames are transmitted correctly
sent = frame_start;

// Adjust the end of the window
int end_frame = frame_start + windowsize;
if (end_frame > total_frames) end_frame = total_frames;

}
}

printf("All frames have been transmitted and acknowledged.\n");

return 0;
}
```

Output:

NAME OF THE EXPERIMENT: 2.c) Sliding window protocol (selective repeat)
OBJECTIVE: Implement the data link layer framing method.
RESOURCE: Code blocks

PROGRAM LOGIC: Selective Repeat ARQ is also known as the Selective Repeat

Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. The Go-back-N ARQ protocol works well if it has fewer errors. But if there is a lot of error in the frame, lots of bandwidth loss in sending the frames again. So, we use the Selective Repeat ARQ protocol. In this protocol, the size of the sender window is always equal to the size of the receiver window.

```c
#include <stdio.h>

#include <stdbool.h>

#define MAX_FRAMES 10

void transmit_frame(int frame) {

    printf("Frame %d has been transmitted.\n", frame);

}

int main() {

    int total_frames, window_size;

    bool ack_received[MAX_FRAMES] = {false};

    int sent_start = 0, sent_end = 0; // Tracks window positions

    int ack;

    // Input for total frames and window size

    printf("Enter the total number of frames to be sent (max %d): ", MAX_FRAMES);

    scanf("%d", &total_frames);

    printf("Enter the window size: ");

    scanf("%d", &window_size);

    if (window_size <= 0 || total_frames <= 0 || window_size > total_frames) {

        printf("Invalid window size or number of frames.\n");

        return 1;

    }

    // Initial transmission of the first window

    for (int i = 0; i < window_size && i < total_frames; i++) {

        transmit_frame(i);

        sent_end++;

    }

    while (sent_start < total_frames) {

        // Receive acknowledgments for the current window
```

```c
        printf("\nPlease enter the acknowledgments received (enter -1 to finish):\n");
    for (int i = sent_start; i < sent_end; i++) {
        if (!ack_received[i]) {
            printf("\nEnter acknowledgment for frame %d: ", i);
            scanf("%d", &ack);
            if (ack == -1) {
                printf("No acknowledgment for frame %d. Retransmitting.\n", i);
                transmit_frame(i);
                printf("Enter acknowledgment for retransmitted frame %d: ", i);
                scanf("%d", &ack);
            }
            if (ack == i) {
                ack_received[ack] = true;
                printf("Acknowledgment %d received.\n", ack);
                // Transmit next frame in sequence if within total frames
                if (sent_end < total_frames) {
                    transmit_frame(sent_end);
                    sent_end++;
                }
            } else {
                printf("Invalid acknowledgment.\n");
            }
        }
    }
    // Slide the window
    while (sent_start < total_frames && ack_received[sent_start]) {
        sent_start++;
    }
}
printf("All frames have been transmitted and acknowledged.\n");
```
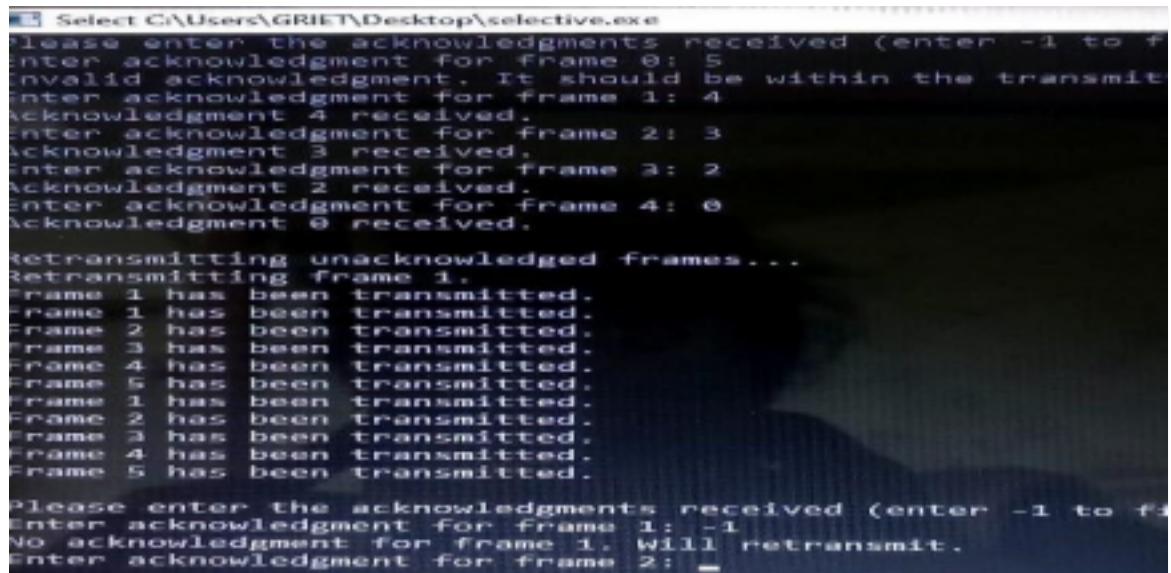
```
    return 0;

}
```

Output::