```
print(f"Ack Receiver for frame {ack_no}")
if ack_no >= start:
    start = ack_no + 1
else
    raise ValueError("Empty acknowledgement line")
except (ValueError, IndexError):
    print(f"Invalid or empty ack-line, resend frame starting
                from {start}")
print("All frames successfully sent")
# Example usage
window_size = int(input("Enter window size:"))
text_message = input("Enter the text message:")
sender(window_size, text_message)
```

RECEIVER_BUFFER.py :

```
import time
def read_sender_Buffer(filename="sender_Buffer.txt"):
    with open(filename,'r') as f:
        frames = f.readlines()

    passed_frame = []
    for line in frames:
        if line.strip():
            try:
                parts = line.strip().split(",")
                frame_no = int(parts[0])
                char = parts[1]
                passed_frames.append(frame_no, char)
            except (IndexError, ValueError):
                print(f"skipping malfunction line: {line}")
    return passed_frames
```

```
def write_receiver_buffer(ack_list, file_name='RECIVER_BUFFER.txt'):
    with open(file_name, 'w') as f:
        for ack in ack_list:
            f.write(f"{ack}, Ack\n")

def receiver():
    expected_frame_no = 0
    while True:
        frames = read_sender_buffer()
        print(f"Received frames: {frames}")
        ack_list = []
        for frame_no, data in frames:
            if frame_no == expected_frame_no:
                print(f"Frame {frame_no} received successfully")
                ack_list.append(expected_frame_no)
                expected_frame_no += 1
            else:
                print(f"Frame {Frame_no} out of order, expecting
                        frames {expected_frame_no}")
                ack_list.append(expected_frame_no-1)
        write_receiver_buffer(ack_list)
        time.sleep(2)

# Main driver
if __name__ == "__main__":
    receiver()
```

OUTPUT:

Sender:

Enter window size : 2

Enter the text message : Hello

Receiver:

Received frames : [(0, 'H'), (1, 'e')]

Frame 0 received successfully

Frame 1 received successfully

Received frames : [(0, 'H'), (1, 'e')]

Frame 0 out of order, expecting frame 2.

Frame 1 out of order, expecting frame 2.

RESULT:

Thus the program for sliding window protocol is successfully executed and output is verified.

9/10/24

2/9/24

# PRACTICAL - 1 : SLIDING WINDOW PROTOCOL

AIM:

Write a program to implement flow control at data link Layer using sliding window protocol. Simulate the flow of frames from one node to another.

CODE:

SENDER.py:

```
import time
import random
def sender (window_size, text_message):
    frames = []
    for i char in enumerate (text_message):
        frames.append ([i, char])
    with open ("sender_Buffer.txt", "w") as file:
        for frame in frames:
            file.write (f"{frame[0]}, {frame[1]}\n")

start = 0
while start < len (frames):
    window = frames [start : start + window_size]
    print (f"sending frames: {window}")

    with open ("sender_Buffer.txt", "w") as file:
        for frame in window:
            file.write (f"{frame[0]}, {frame[1]}\n")
    time.sleep (1)

try:
    with open ("Receiver_Buffer.txt", "r") as file:
        ack_line = file.readline().strip()
    if ack_line
        ack_no = int (ack_line.split (",") [0])
```