a. Header Section
■ Shruti Jayavardhanan
■ Submission Date - 03/31/2024
■ Coding Language - Python
■ Submission file names: mainGBN.py, GBN_Sender.py, GBN_Receiver.py, packet.py, msg.py, simulator.py, circularbuffer.py, event.py, event_list.py

b. Compilation Section
- Ensure Python is installed on your system.
- Navigate to the project directory using the command line or terminal.

c. Execution Section
- To execute the project, set the type to 'GBN' in mainGBN.py and then enter "python mainGBN.py" in the command prompt or terminal.

d. Description Section
- The Go-Back-N (GBN) protocol is a sliding window protocol used for reliable data transfer in computer networks. In this implementation, the GBN protocol is used to transfer data from a sender to a receiver.
- Sender:
    - The sender (GBN_Sender.py) waits for messages from the application layer (layer 5) in the "WAIT_LAYER5" state.
    - Upon receiving a message from layer 5 (S_output function), the sender creates packets with sequence numbers and sends them to the receiver via layer 3.
    - After sending packets, the sender waits for ACKs from the receiver in the "WAIT_ACK" state.
    - Upon receiving ACKs, the sender updates the window and sends new packets or retransmits packets as necessary.
- Receiver:
    - The receiver (GBN_Receiver.py) waits for packets from layer 3.
    - Upon receiving packets from layer 3 (R_input function), the receiver verifies the checksum and the sequence number to ensure packet integrity and correctness.
    - If the received packet is valid, the receiver sends an ACK to the sender with the acknowledgment number. The receiver then updates its expected sequence number.
    - If the received packet is corrupted or out of order, it is discarded, and appropriate counters are updated.
- Functions Implemented:
    - S_output(message): Handles the message received from layer 5. It constructs packets with messages and sends them to layer 3.
    - S_input(received_packet): Processes packets received from layer 3. It verifies ACK numbers and handles them accordingly.
    - S_handle_timer(): Manages the expiration of the sender's timer. It retransmits packets if no ACKs are received within the timeout period.

    - R_input(received_packet): Deals with packets received from layer 3 at the receiver's end. It verifies packet integrity, sends ACKs for correct packets, and updates the sequence number.
    - Initialization functions: Initializes relevant class variables and states for both sender and receiver.
- Additional Data Structures/Fields/Methods:
    - Circular Buffer: Used for storing outstanding and unacknowledged packets at the sender's end.
    - Event List: Manages events in the simulation, such as timeouts and packet arrivals.
    - Packet and Message Classes: Represent packets and messages exchanged between sender and receiver.