SIT – 202 COMPUTER NETWORKS AND COMMUNICATION :

DNS SERVER SKETCH

PSEUDOCODE FOR DNS SERVER:

```
# Constants
DEFINE DNS_PORT AS 53
DEFINE BUFFER_SIZE AS 512

# Data storage for DNS records
DEFINE A_RECORDS AS {
   "example.com": "93.184.216.34"
}
DEFINE CNAME_RECORDS AS {
   "www.example.com": "example.com"
}

# Function to initialize the DNS server
FUNCTION initialize_dns_server()
   # Create and bind a UDP socket
   server_socket = CREATE_UDP_SOCKET()
   BIND_SOCKET(server_socket, DNS_PORT)

   # Load DNS records (if needed)
   LOAD_A_RECORDS()
   LOAD_CNAME_RECORDS()

   PRINT "DNS Server initialized and listening for queries"
   RETURN server_socket
END FUNCTION
```

```
# Function to create a UDP socket
FUNCTION CREATE_UDP_SOCKET()
    RETURN CREATE_SOCKET(AF_INET, SOCK_DGRAM)
END FUNCTION

# Function to bind a socket to a port
FUNCTION BIND_SOCKET(socket, port)
    BIND(socket, '', port)
END FUNCTION

# Function to load A records (placeholder)
FUNCTION LOAD_A_RECORDS()
    # Logic to load A records from a file or database
END FUNCTION

# Function to load CNAME records (placeholder)
FUNCTION LOAD_CNAME_RECORDS()
    # Logic to load CNAME records from a file or database
END FUNCTION

# Function to listen for incoming queries
FUNCTION LISTEN_FOR_QUERIES(server_socket)
    WHILE TRUE
        # Receive data from the client
        data, client_address = RECEIVE_DATA(server_socket)

        # Parse the query to extract hostname and query type
        hostname, query_type = PARSE_QUERY(data)
```

```
        # Process the query based on type
        IF query_type IS "A"
            HANDLE_A_RECORD(hostname, client_address, server_socket)
        ELSE IF query_type IS "CNAME"
            HANDLE_CNAME_RECORD(hostname, client_address, server_socket)
        ELSE
            SEND_ERROR_RESPONSE(client_address, server_socket)
        END IF
    END WHILE
END FUNCTION

# Function to receive data from the socket
FUNCTION RECEIVE_DATA(socket)
    RETURN RECEIVE_FROM(socket, BUFFER_SIZE)
END FUNCTION

# Function to parse the query
FUNCTION PARSE_QUERY(data)
    # Decode and extract hostname and query type
    decoded_data = DECODE(data)
    hostname = EXTRACT_HOSTNAME(decoded_data)
    query_type = DETERMINE_QUERY_TYPE(decoded_data)

    RETURN hostname, query_type
END FUNCTION

# Function to handle A record queries
FUNCTION HANDLE_A_RECORD(hostname, client_address, server_socket)
    IF hostname IN A_RECORDS
```

```
        ip_address = A_RECORDS[hostname]
        response = GENERATE_DNS_RESPONSE(hostname, "A", ip_address)
    ELSE
        response = "Error: A record not found"
    END IF

    SEND_RESPONSE(response, client_address, server_socket)
END FUNCTION

# Function to handle CNAME record queries
FUNCTION HANDLE_CNAME_RECORD(hostname, client_address, server_socket)
    IF hostname IN CNAME_RECORDS
        canonical_name = CNAME_RECORDS[hostname]
        response = GENERATE_DNS_RESPONSE(hostname, "CNAME", canonical_name)
    ELSE
        response = "Error: CNAME record not found"
    END IF

    SEND_RESPONSE(response, client_address, server_socket)
END FUNCTION

# Function to generate DNS response
FUNCTION GENERATE_DNS_RESPONSE(hostname, record_type, data)
    IF record_type IS "A"
        RETURN "Hostname: " + hostname + ", Type: A, IP: " + data
    ELSE IF record_type IS "CNAME"
        RETURN "Hostname: " + hostname + ", Type: CNAME, Canonical Name: " + data
    ELSE
        RETURN "Error: Unsupported record type"
```

```
    END IF
END FUNCTION

# Function to send response to the client
FUNCTION SEND_RESPONSE(response, client_address, server_socket)
    SEND_TO(server_socket, response, client_address)
END FUNCTION

# Function to send error response
FUNCTION SEND_ERROR_RESPONSE(client_address, server_socket)
    response = "Error: Query type not supported"
    SEND_RESPONSE(response, client_address, server_socket)
END FUNCTION

# Main function to run the DNS server
FUNCTION MAIN()
    server_socket = INITIALIZE_DNS_SERVER()
    LISTEN_FOR_QUERIES(server_socket)
END FUNCTION

# Run the DNS server
MAIN()
```