# תרגיל 2
## הגשה ביחידים

# PART A: Learn to use datagram sockets by example.

Read and understand programs **Recv_udp.c** and **Send_udp.c**. Compile each program (**C9**) and execute them both in your local host using the following instructions:

1. Run **Recv_udp** (note that this program contains an infinite loop, so you will want to add '**&**' to the execution command so the shell will return to the prompt, or just open another terminal window).

2. Run **Send_udp** several times, passing **localhost** as command line parameter, and observe what happens.

3. Go back to the source code and do your best to comment **each line** (in the files **Recv_udp.c** and **Send_udp.c**), specifically state where a **system call** is issued and explain what the line accomplishes. Your comments should be short and straight to the point.

4. Write the code for a new function called **printsin()** according to the specifications below. uncomment the calls to **printsin()** in **Recv_udp.c** and build the executable.

   The function prototype must be:
   **void printsin(struct sockaddr_in *sin, char *pname, char* msg)**

   The function will print the string indicated by *pname* ended with a line break, then it will print the string indicated by *msg*, then the string "ip= " follow by the IP address of the host associated with this socket address structure in dotted-decimal notation, a comma, and finally the string "port= " followed by the port number associated with this socket address structure. When this function is called, it should produce output as in the examples below:

   <u>function call:</u> **printsin(&s_in, "UDP-SERVER:", "Local socket is:");**
   <u>output:</u>
   **UDP-SERVER:**
   **Local socket is: ip= 0.0.0.0, port= 9000**

   <u>function call:</u> **printsin((struct sockaddr_in*)&from, "UDP-SERVER: ", "Packet from:");**
   **UDP-SERVER:**
   **Packet from: ip= 127.0.0.1, port= 33080**

5. Change the server and client code as following: The client will send your name to the server and then will receive the server response. On the server side, the server

will receive the client name (instead of the current msg structure) and will send back its name. (make sure to given a distinct name to both)

6. Copy the server and client code to new files named **Server.c , Client.c, Router.c** and change them as following:
   **Motivation:** Node A and Node C communicate via node B (the router).

   **Requirements**:
   When Node B receives a message from Node A:
   a. B chooses a random number uniformly from range [0,1].
   b. If the number is greater than X it B will forward the message, otherwise, it will drop the message (delete it).

   On the vice versa, B always forwards any message from C.

   -where X is an argument received from the command prompt.

   **Important Note:**

   Make sure Node A and Node C can message each other from the command prompt freely, and when one side wants to disconnect he can do it by typing **exit**