University Of Central Punjab

Faculty of Information Technology

FINAL TERM EXAMINATION

Course Title: Computer Communications and Networks Lab [CSNC2411]	**Semester: Fall 2022**
Time Allowed: 90 Minutes (1.5 HOURS)	**Total Marks: 50**

Name: _____Reg No: _____ Sec__

Instructions

1. Write your Name, Registration number and section on the Word file.
2. Submit only one docx file.
3. Attach the screenshot and code where required.
4. Calculators are not allowed.
5. Attempt all questions.
6. There are two submissions; one for windows and another for Ubuntu.

Q1. **You are required to create an concurrent TCP client-server connection with the following functionality. (Helping material code with all relevant networking-calls syntaxes and libraries is given at the end of the sheet) [25]**

Write a C program that takes a sentence as input and prints the longest word in it.

Client 1

1- The Client will ask the user to type any string (e.g How are you Tariq)

2- The Client will send this string to the Server.

Client 2

1- The Client will ask the user to type any string (e.g Welcome to final exam)

2- The Client will send this string to the Server.

Server side:

1- The Server will initially wait for the Client to receive string

2- After receiving the string, the Server will display the longest word in the both string

e.g(Longest word is : Welcome)

Q. No.2**: Packet Tracer**

- Attach the screenshot below the highlighted questions [25]

Packet Tracer - Implement Basic Connectivity

Addressing Table

Device	Interface	IP Address	Subnet Mask
S1		192.168.1.253	255.255.255.0
S2	VLAN 1	192.168.1.254	255.255.255.0
PC1	NIC	192.168.1.1	255.255.255.0
PC2	NIC	192.168.1.2	255.255.255.0

Objectives

Part 1: Perform a Basic Configuration on S1

Part 2: Configure the PCs

Part 3: Configure the Switch Management Interface

Background

In this activity you will first perform basic switch configurations. Then you will implement basic connectivity by configuring IP addressing on switches and PCs. When the IP addressing configuration is complete, you will use various **show** commands to verify configurations and use the **ping** command to verify basic connectivity between devices.

Part 1: Perform a Basic Configuration on S1

Complete the following steps on S1

Step 1: Configure S1 with a hostname.

a. Click **S1**, and then click the **CLI** tab.

b. Enter the correct command to configure the hostname as **S1**.

Step 2: Configure the console and privileged EXEC mode passwords.

a. Use **cisco** for the console password.

b. Use **class** for the privileged EXEC mode password.

Step 3: Verify the password configurations for S1.

How can you verify that both passwords were configured correctly?

Step 4: Configure a message of the day (MOTD) banner.

Use an appropriate banner text to warn unauthorized access. The following text is an example:

Authorized access only. Violators will be prosecuted to the full extent of the law.

Step 5: Save the configuration file to NVRAM.

Which command do you issue to accomplish this step?

Part 2: Configure the PCs

Configure PC1 with IP addresses.

Step 1: Configure both PCs with IP addresses.

a. Click **PC1**, and then click the **Desktop** tab.

b. Click **IP Configuration**. In the **Addressing Table** above, you can see that the IP address for PC1 is 192.168.1.1 and the subnet mask is 255.255.255.0. Enter this information for PC1 in the **IP Configuration** window.

Step 2: Test connectivity to switches.

a. Click **PC1**. Close the **IP Configuration** window if it is still open. In the **Desktop** tab, click **Command Prompt**.

b. Type the **ping** command and the IP address for S1, and press **Enter**.

```
Packet Tracer PC Command Line 1.0
PC> ping 192.168.1.253
```

Were you successful? Why or why not?

Part 3: Configure the Switch Management Interface

Configure S1 with an IP address.

Step 1: Configure S1 with an IP address.

Switches can be used as a plug-and-play device, meaning they do not need to be configured for them to work. Switches forward information from one port to another based on Media Access Control (MAC) addresses. If this is the case, why would we configure it with an IP address?

Use the following commands to configure S1 with an IP address.

```
S1 #configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)# interface vlan 1
S1(config-if)# ip address 192.168.1.253 255.255.255.0
S1(config-if)# no shutdown
%LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state
to up
S1(config-if)#
S1(config-if)# exit
S1#
```

Why do you need to enter the **no shutdown** command?

Step 2: Verify the IP address configuration on S1

Use the **show ip interface brief** command to display the IP address and status of the all the switch ports and interfaces. Alternatively, you can also use the **show running-config** command.

Step 4: Save configurations for S1 to NVRAM.

Which command is used to save the configuration file in RAM to NVRAM?

Step 5: Verify network connectivity.

Network connectivity can be verified using the **ping** command. It is very important that connectivity exists throughout the network. Corrective action must be taken if there is a failure. Ping S1's IP address from PC1.

a. Click **PC1**, and then click the **Desktop** tab.

b. Click **Command Prompt**.

d. Ping the IP address for S1.

Code:

In the client side add the ip address (127.0.0.1) at the time of run
e.g gcc echoClient.c -o echoClient
./echoClient 127.0.0.1

Server side:
gcc echoServer.c -o echoServer
./echoServer

Client Code:

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>


#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/

int
main(int argc, char **argv)
{
 int sockfd;
 struct sockaddr_in servaddr;
 char sendline[MAXLINE], recvline[MAXLINE];

 //basic check of the arguments
 //additional checks can be inserted
 if (argc !=2) {
  perror("Usage: TCPClient <IP address of the server");
  exit(1);
 }

 //Create a socket for the client
 //If sockfd<0 there was an error in the creation of the socket
 if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
  perror("Problem in creating the socket");
  exit(2);
 }
```

```
//Creation of the socket
memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr= inet_addr(argv[1]);
servaddr.sin_port =  htons(SERV_PORT); //convert to big-endian order

//Connection of the client to the socket
if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr))<0) {
 perror("Problem in connecting to the server");
 exit(3);
}

while (fgets(sendline, MAXLINE, stdin) != NULL) {

 send(sockfd, sendline, strlen(sendline), 0);

 if (recv(sockfd, recvline, MAXLINE,0) == 0){
  //error: server terminated prematurely
  perror("The server terminated prematurely");
  exit(4);
 }
 printf("%s", "String received from the server: ");
 fputs(recvline, stdout);
}

 exit(0);
}
```

Server Code:

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>
#include <unistd.h>


#define MAXLINE 4096 /*max text line length*/
#define SERV_PORT 3000 /*port*/
#define LISTENQ 8 /*maximum number of client connections*/

int main (int argc, char **argv)
```

```
{
 int listenfd, connfd, n;
 pid_t childpid;
 socklen_t clilen;
 char buf[MAXLINE];
 struct sockaddr_in cliaddr, servaddr;

 //Create a socket for the soclet
 //If sockfd<0 there was an error in the creation of the socket
 if ((listenfd = socket (AF_INET, SOCK_STREAM, 0)) <0) {
  perror("Problem in creating the socket");
  exit(2);
 }


 //preparation of the socket address
 servaddr.sin_family = AF_INET;
 servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
 servaddr.sin_port = htons(SERV_PORT);

 //bind the socket
 bind (listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));

 //listen to the socket by creating a connection queue, then wait for clients
 listen (listenfd, LISTENQ);

 printf("%s\n","Server running...waiting for connections.");

 for ( ; ; ) {

  clilen = sizeof(cliaddr);
  //accept a connection
  connfd = accept (listenfd, (struct sockaddr *) &cliaddr, &clilen);

  printf("%s\n","Received request...");

  if ( (childpid = fork ()) == 0 ) {//if it's 0, it's child process

    printf ("%s\n","Child created for dealing with client requests");

    //close listening socket
    close (listenfd);

    while ( (n = recv(connfd, buf, MAXLINE,0)) > 0)  {
     printf("%s","String received from and resent to the client:");
```

```
     puts(buf);
     send(connfd, buf, n, 0);
    }

    if (n < 0)
      printf("%s\n", "Read error");
    exit(0);
  }
  //close socket of the server
  close(connfd);
 }
}
```