Q. Write a socket program in C to create a chat between client and server in TCP

Firsst client

```
#include<stdio.h>
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr

void func(int sockfd)
{
char buff[MAX];
int n;
for(;;)
{
bzero(buff,sizeof(buff));
printf("Enter the string : ");
n=0;
while((buff[n++]=getchar())!='\n');
write(sockfd,buff,sizeof(buff));
bzero(buff,sizeof(buff));
read(sockfd,buff,sizeof(buff));
printf("From Server : %s",buff);
if((strncmp(buff,"exit",4))==0)
{
printf("Client Exit...\n");
break;
}
```

```
}
}
int main()
{
int sockfd,connfd;
struct sockaddr_in servaddr,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd==-1)
{
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
servaddr.sin_port=htons(PORT);
if(connect(sockfd,(SA *)&servaddr,sizeof(servaddr))!=0)
{
printf("connection with the server failed...\n");
exit(0);
}
else
printf("connected to the server..\n");
func(sockfd);
close(sockfd);
}
```

```
#include<stdio.h>
```

```
#include<netinet/in.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netdb.h>
#include<string.h>
#define MAX 80
#define PORT 43454
#define SA struct sockaddr

void func(int sockfd)
{
char buff[MAX];
int n;
for(;;)
{
 bzero(buff,MAX);
 read(sockfd,buff,sizeof(buff));
 printf("From client: %s\t To client : ",buff);
 bzero(buff,MAX);
 n=0;
 while((buff[n++]=getchar())!='\n');
 write(sockfd,buff,sizeof(buff));
 if(strncmp("exit",buff,4)==0)
{
printf("Server Exit...\n");
break;
}
}
}
int main()
{
```

```
int sockfd,connfd,len;
struct sockaddr_in servaddr,cli;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd==-1)
{
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
servaddr.sin_port=htons(PORT);
if((bind(sockfd,(SA *)&servaddr,sizeof(servaddr)))!=0)
{
printf("socket bind failed...\n");
exit(0);
}
else
printf("Socket successfully binded..\n");
if((listen(sockfd,5))!=0)
{
printf("Listen failed...\n");
exit(0);
}
else
printf("Server listening..\n");
len=sizeof(cli);
connfd=accept(sockfd,(SA *)&cli,&len);
if(connfd<0)
```

```
{
printf("server acccept failed...\n");
exit(0);
}
else
printf("server acccept the client...\n");
func(connfd);
close(sockfd);
}
```

Write a socket program in C to create a chat between client and server in UDP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAX 100
#define SA struct sockaddr
#define PORT 8080

int main()
{
  int sockfd, len;
  struct sockaddr_in server_addr;
  char server_msg[MAX], client_msg[MAX];
```

```
// creating the new socket

sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0)
{
  printf("Error in socket creation..\n");
  exit(0);
}
else
  printf("Socket created successfully..\n");

bzero(&server_addr, sizeof(server_addr));// set to zero value

// assigning IP and PORT
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

while (1)
{
  len = sizeof(server_addr);
  memset(client_msg, '\0', sizeof(client_msg));
  memset(server_msg, '\0', sizeof(server_msg));

  // Sending message to client
  printf("To Server : ");
  gets(client_msg);
  sendto(sockfd, client_msg, sizeof(client_msg), 0, (SA *)&server_addr, sizeof(server_addr));
  if (strcmp(client_msg, "exit") == 0)
  {
    printf("Closing the chat...\n");
```

```
    break;
   }

   // receiving message from client
   recvfrom(sockfd, server_msg, sizeof(server_msg), 0, (SA *)&server_addr, &len);
   printf("\tFrom Server : %s\n", server_msg);
   if (strcmp(server_msg, "exit") == 0)
   {
    printf("Server Exit...\n");
    break;
   }
  }
  close(sockfd);
  return 0;
}


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>//Provides functions to manipulate internet addresses, like inet_addr().
#define MAX 1000
#define SA struct sockaddr
#define PORT 8080

int main()
{
```

```
int sockfd, len;
struct sockaddr_in server_addr, client_addr;
char server_msg[MAX], client_msg[MAX];

// creating the new socket

sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if (sockfd < 0)
{
  printf("Error in socket creation..\n");
  exit(0);
}
else
  printf("Socket created successfully..\n");

bzero(&server_addr, sizeof(server_addr));
bzero(&client_addr, sizeof(client_addr));

// assigning IP and PORT
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);

if (bind(sockfd, (SA *)&server_addr, sizeof(server_addr)) != 0)
{
  printf("Socket binding failed...\n");
  exit(0);
}
else
  printf("Socket successfully binded..\n");
```

```
  printf("Listening for incoming messages...\n\n");


  while (1)
  {
    len = sizeof(client_addr);


    memset(client_msg, '\0', sizeof(client_msg));
    memset(server_msg, '\0', sizeof(server_msg));


    // receiving message from client
    recvfrom(sockfd, client_msg, sizeof(client_msg), 0, (SA *)&client_addr, &len);// socket ,buffer, buffer size, 0 is default,struct addres,and length
    printf("From Client : %s\n", client_msg);
    if(strcmp(client_msg,"exit")==0){
      printf("Client Exit...\n");
      break;
    }


    // Sending message to client
    printf("\tTo Client : ");
    gets(server_msg);
    sendto(sockfd, server_msg, sizeof(server_msg), 0, (SA *)&client_addr, len);// socket,message, size,0,struct address,length
    if(strcmp(server_msg,"exit")==0){
      printf("Closing the chat...\n");
      break;
    }
  }
  close(sockfd);
  return 0;
}
```

Write a socket program in C to transfer a file from the client to server in TCP.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 15050
#define MAX_BUFFER_SIZE 1024

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[MAX_BUFFER_SIZE];
    FILE* fp;

    // Create socket
    client_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (client_fd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
    // Connect to the server
    if (connect(client_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection to server failed");
        exit(EXIT_FAILURE);
    }

    // Open a file for writing
    fp = fopen("received_file.txt", "w");
    if (fp == NULL) {
        perror("File open failed");
        exit(EXIT_FAILURE);
    }

    while (1) {
        int n = read(client_fd, buffer, MAX_BUFFER_SIZE); // Receive data from the server
        if (n <= 0) {
            break;
        }
        fwrite(buffer, 1, n, fp); // Write the data to the file
    }

    printf("File received successfully.\n");

    close(client_fd);
    fclose(fp);

    return 0;
}



#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 15050
#define MAX_BUFFER_SIZE 1024

int main() {
    int server_fd, new_socket;
    struct sockaddr_in server_addr, new_addr;
    socklen_t addr_size;
    char buffer[MAX_BUFFER_SIZE];
    FILE* fp;

    // Create socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (server_fd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;

    // Bind the socket
    if (bind(server_fd, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("Binding failed");
        exit(EXIT_FAILURE);
    }
```

```
// Listen for incoming connections
if (listen(server_fd, 5) < 0) {
   perror("Listen failed");
   exit(EXIT_FAILURE);
}

printf("Server listening on port %d...\n", PORT);
addr_size = sizeof(new_addr);

// Accept connection from client
new_socket = accept(server_fd, (struct sockaddr*)&new_addr, &addr_size);

// Open the file for reading
fp = fopen("demo.txt", "rb"); // Use "rb" mode for reading binary files
if (fp == NULL) {
   perror("File open failed");
   exit(EXIT_FAILURE);
}

while (1) {
   // Read from file into buffer
   int n = fread(buffer, 1, MAX_BUFFER_SIZE, fp);
   if (n > 0) {
      write(new_socket, buffer, n); // Send the data to the client
   }
   if (n < MAX_BUFFER_SIZE) {
      if (feof(fp)) {
         printf("File sent successfully.\n");
      }
      if (ferror(fp)) {
```

```
            perror("Error reading from file");
        }
        break;
      }
   }

   close(new_socket);
   close(server_fd);
   fclose(fp);

   return 0;
}
```

Write a socket program in C to transfer a file from the client to server in UDP.

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

#define MAX 200
#define PORT 43454
#define SA struct sockaddr

void func(int sockfd, struct sockaddr_in servaddr) {
```

```
    char buff[MAX];
    FILE *fp;
    int n;
    socklen_t len = sizeof(servaddr);

    // Clear the buffer and open the file to send
    bzero(buff, sizeof(buff));
    fp = fopen("clifile", "r");
    if (fp == NULL) {
        perror("File open failed");
        exit(EXIT_FAILURE);
    }

    // Read the file into the buffer
    n = 0;
    while ((buff[n++] = getc(fp)) != EOF);

    // Send the file contents to the server
    sendto(sockfd, buff, sizeof(buff), 0, (SA *)&servaddr, len);
    printf("File sent to the server.\n");

    fclose(fp);
}

int main() {
    int sockfd;
    struct sockaddr_in servaddr;

    // Create a UDP socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
```

```
        printf("Socket creation failed...\n");
        exit(0);
    } else {
        printf("Socket successfully created..\n");
    }

    // Clear and set up the server address struct
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    servaddr.sin_port = htons(PORT);

    // Call the function to send the file
    func(sockfd, servaddr);

    // Close the socket
    close(sockfd);
    return 0;
}



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

#define MAX 80
#define PORT 43454
```

```
#define SA struct sockaddr

void func(int sockfd)
{
    char buff[MAX];
    struct sockaddr_in cliaddr;
    socklen_t len = sizeof(cliaddr);
    FILE *fp;

    bzero(buff, MAX);

    // Receive message from client
    int n = recvfrom(sockfd, buff, sizeof(buff), 0, (SA*)&cliaddr, &len);
    if (n < 0) {
        perror("Error in receiving data");
        exit(1);
    }
    printf("From client: %s\n", buff);

    // Write message to file
    fp = fopen("sfile", "a");
    if (fp == NULL) {
        perror("Error opening file");
        exit(1);
    }
    fputs(buff, fp);
    fclose(fp);
}

int main()
{
```

```
    int sockfd;
    struct sockaddr_in servaddr, cliaddr;

    // Create UDP socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == -1) {
        printf("Socket creation failed...\n");
        exit(0);
    } else
        printf("Socket successfully created..\n");

    // Clear and set up server address structure
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(PORT);

    // Bind the socket to the specified port
    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
        printf("Socket bind failed...\n");
        exit(0);
    } else
        printf("Socket successfully binded..\n");

    // Run function to receive message and save it to a file
    func(sockfd);

    // Close the socket
    close(sockfd);
}
```

Write a socket program in C to handle multiple clients using fork in TCP.

```
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#define SERV_TCP_PORT 5035

int main(int argc, char* argv[])
{
   int sockfd;
   struct sockaddr_in serv_addr;
   char buffer[4096];

   while (strcmp(buffer, "Quit") != 0)
   {
      sockfd = socket(AF_INET, SOCK_STREAM, 0);
      serv_addr.sin_family = AF_INET;
      serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
      serv_addr.sin_port = htons(SERV_TCP_PORT);

      printf("\nReady for sending\n");
      connect(sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

      fgets(buffer, 4096, stdin);  // Fixed buffer size
      write(sockfd, buffer, 4096); // Fixed buffer size
   }

   close(sockfd);
   return 0;
```

```
}



#include<stdio.h>
#include<arpa/inet.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<netdb.h>
#define SERV_TCP_PORT 5035
int main(int argc,char**argv)
{
      int i,sockfd,newsockfd,clength;
      struct sockaddr_in serv_addr,cli_addr;
      char buffer[4098];
      pid_t childpid;
      sockfd=socket(AF_INET,SOCK_STREAM,0);
      serv_addr.sin_family=AF_INET;
      serv_addr.sin_addr.s_addr=INADDR_ANY;
      serv_addr.sin_port=htons(SERV_TCP_PORT);
      printf("\nStart");
      bind(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));
      printf("\nListening...\n");
      listen(sockfd,5);
      i=1;
      for(;;)
      {
          clength=sizeof(cli_addr);
         listen(sockfd,5);
         newsockfd=accept(sockfd,(struct sockaddr*)&cli_addr,&clength);
```

```
        printf("\n\nAccepted");
        if((childpid=fork())==0)
        {
            printf("\nChild PID is : %d",getpid());
            read(newsockfd,buffer,4096);
            printf("\nClient Message : %s",buffer);
        }
        close(newsockfd);
        i++;
    }
    return 0;
}
```

Write a socket program in C to determine the IP address from a given hostname / URL.

```
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <sys/types.h>
#include <arpa/inet.h>

int main()
{

int i;
char host[80];
struct hostent *he;
struct in_addr addr;
```

```
   printf("Enter the host-name :" );
   gets(host);
   he = gethostbyname(host);
   if (he == NULL)
   {
      printf("The address cannot be resolved at this time");
      exit(1);
   }

   printf("Official name is: %s\n", he->h_name);
   printf("IP address: %s\n", inet_ntoa(*(struct in_addr*)he->h_addr));

}

//Official name is: star-mini.c10r.facebook.com .....star-mini.c10r (host
name).....facebook.com (domain name)




#include <stdio.h> //for printf()
#include <stdlib.h> //for exit()
#include <arpa/inet.h> //for inet_pton()
#include <netdb.h> // for NI_MAXHOST, getnameinfo() and gai_strerror()
#include <errno.h> // for errno
#include <string.h> // for strerror()

int main(int argc, char** argv) {
  if(argc<2) {
   printf("\n%s [IP]\n",argv[0]);
   printf("For e.g. %s 10.32.129.77\n",argv[0]);
   exit(-1);
```

```
}
struct sockaddr_in sa;
int res = inet_pton(AF_INET, argv[1] , &sa.sin_addr);
switch(res) {
  case 0: printf("\nInput address is not a valid IPv4 address.\n");
  case -1: if(res == -1)
            printf("\nError(%s)\n",strerror(errno));
          int n_res = inet_pton(AF_INET6, argv[1] , &sa.sin_addr);
          switch(n_res) {
            case 0: printf("\nInput address is not a valid IPv6 address.\n");
            case -1: if(n_res == -1)
                      printf("\nError(%s)\n",strerror(errno));
                     exit(-1);
            case 1: sa.sin_family = AF_INET6;
          }
  case 1: sa.sin_family = AF_INET;
}

printf("\nsa.sin_addr.s_addr[%d]\n",sa.sin_addr.s_addr);

char node[NI_MAXHOST];
memset(node,0,NI_MAXHOST);
res = getnameinfo((struct sockaddr*)&sa, sizeof(sa), node, sizeof(node), NULL, 0, 0);
if (res) {
  printf("%s\n", gai_strerror(res));
  exit(1);
}

printf("\nIP[%s]\n",argv[1]);
printf("HOSTNAME[%s]\n", node);
```

```
  return 0;
}
// example: ./filename 157.240.1.35 (run along with IP address)
```

Write a socket program in C to provide the current Time and Date to the client in TCP.

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

#define MAX 30
#define PORT 8080
#define SA struct sockaddr

int main(){
   int sockfd;
   struct sockaddr_in server_addr;
   char buff[MAX];

   sockfd = socket(AF_INET,SOCK_STREAM,0);

   if(sockfd<0){
      printf("Error in socket creation.\n");
      exit(0);
   }
```

```
    server_addr.sin_family=AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    connect(sockfd, (SA*)&server_addr, sizeof(server_addr));
    recv(sockfd,buff,MAX,0);
    printf("\nTIME FROM SERVER %s\n",buff);
    close(sockfd);
    return 0;
}


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

#define MAX 30
#define PORT 8080
#define SA struct sockaddr

// server_addr-> having all value of server
// client_addr-> having all value of client
int main(){
    int sockfd,connfd;
    struct sockaddr_in server_addr,client_addr;
    time_t curr_time;
```

```
    time(&curr_time);
    int countClient = 0;
    sockfd = socket(AF_INET,SOCK_STREAM,0);// internet Adrress concept use,i would i like to use tcp concept=> then created socket

    if(sockfd<0){
        printf("Error in socket creation.\n");
        exit(0);
    }
    else
        printf("Server created successfully.\n");

    server_addr.sin_family=AF_INET;// internet address
    server_addr.sin_port = htons(PORT);// convert integer into network format

    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(sockfd,(SA*)&server_addr,sizeof(server_addr)) !=0 ){
        printf("Binding failed\n");
        exit(0);
    }
    else
        printf("Socket binded successfully\n");

    if(listen(sockfd,10) != 0){// how many client handle for this example 10 client handle
        printf("Listening failed\n");
        exit(0);
    }
    else
        printf("Server Listening.\n");

    while(1){
```

```
        countClient++;
        int len = sizeof(client_addr);
        connfd = accept(sockfd, (SA*)&client_addr, &len);// now i have to accept new socket created for clinet

        // char *string = asctime(timeinfo);
        time(&curr_time);
        printf("\nClient %d has requested for time at %s", countClient, ctime(&curr_time));

        //sending time to the client side
        // ctime(&time_from_pc)
        time(&curr_time);
        send(connfd,ctime(&curr_time), MAX, 0);
    }

    return 0;
}
```

Write a socket program in C to provide the current Time and Date to the client in UDP.

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

```
#define MAX 30
#define PORT 8080
#define SA struct sockaddr

int main(){
   int sockfd;
   struct sockaddr_in server_addr;
   char buff[MAX];

   sockfd = socket(AF_INET,SOCK_DGRAM,0);

   if(sockfd<0){
      printf("Error in socket creation.\n");
      exit(0);
   }

   server_addr.sin_family=AF_INET;
   server_addr.sin_port = htons(PORT);
   server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
   int len = sizeof(server_addr);

   strcpy(buff,"");
   sendto(sockfd,buff,MAX,0,(SA*)&server_addr,len);
   recvfrom(sockfd,buff,MAX,0,(SA*)&server_addr,&len);
   printf("\nTime From Server : %s\n",buff);
   close(sockfd);
   return 0;
}
```

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

#define MAX 30
#define PORT 8080
#define SA struct sockaddr

int main(){
   int sockfd;
   struct sockaddr_in server_addr,client_addr;
   time_t curr_time;
   time(&curr_time);
   int countClient = 0;
   char buff[MAX];
   sockfd = socket(AF_INET,SOCK_DGRAM,0);

   if(sockfd<0){
      printf("Error in socket creation.\n");
      exit(0);
   }
   else
      printf("Server created successfully.\n");
   bzero(&server_addr,sizeof(server_addr));
```

```
    server_addr.sin_family=AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if(bind(sockfd,(SA*)&server_addr,sizeof(server_addr)) !=0 ){
        printf("Binding failed\n");
        exit(0);
    }
    else
        printf("Socket binded successfully\n");

    while(1){
        countClient++;
        int len = sizeof(client_addr);
        recvfrom(sockfd,buff,MAX,0,(SA*)&client_addr,&len);
        // char *string = asctime(timeinfo);
        time(&curr_time);
        printf("\nClient %d has requested for time at %s", countClient, ctime(&curr_time));

        //sending time to the client side
        // ctime(&time_from_pc)

        time(&curr_time);
        sendto(sockfd,ctime(&curr_time), MAX, 0,(SA*)&client_addr,len);

    }

    return 0;
}
```

Create a LAN using one Router and two Switch and simulate it.

Simulate LAN using one Router and three Switch.

Connect two network using one Router and one Hub and one Switch.

Simulate the working of Hub and Switch for a network, and show the Mac address table.