Q)1 way multiple time communication

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
void main(){
  int fd[2], pid=0;
  char message[1024], buffer[1024];
  if(pipe(fd)==-1){
    perror("PIPE CREATION FAILED\n");
    exit(0);
  }
  pid=fork();
  if(pid>0){
    close(fd[0]);
    while(1){
      memset(message, sizeof(message), 0);
      gets(message);
      write(fd[1], message, 1024);
      if(strcmp(message,"exit")==0) break;
    }
  }
  else{
    close(fd[1]);
    while(1){
      memset(buffer, sizeof(buffer), 0);
      read(fd[0],buffer,1024);
      printf("%s\n",buffer);
      if(strcmp(buffer, "exit")==0) break;
    }
  }
}
```

Q)2 way multiple time communication

```
#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

void main(){

  int fd1[2], fd2[2], pid=0;

  char message[1024], buffer[1024];

  if(pipe(fd1)==-1){

    perror("PIPE CREATION FAILED\n");

    exit(0);

  }

  if(pipe(fd2)==-1){

    perror("PIPE CREATION FAILED\n");

    exit(0);

  }

  pid=fork();

  if(pid>0){

    close(fd1[0]);

    close(fd2[1]);

    while(1){

      memset(message, sizeof(message), 0);

      printf("Enter data for child: ");

      gets(message);

      write(fd1[1], message, 1024);

      if(strcmp(message,"exit")==0) break;

      memset(buffer, sizeof(buffer), 0);

      read(fd2[0],buffer,1024);

      printf("Received data from child: %s\n",buffer);

      if(strcmp(buffer, "exit")==0) break;

    }

  }
```

```
    else{
        close(fd1[1]);
        close(fd2[0]);
        while(1){
            memset(buffer, sizeof(buffer), 0);
            read(fd1[0],buffer,1024);
            printf("Received data from parent: %s\n",buffer);
            if(strcmp(buffer, "exit")==0) break;
            memset(message, sizeof(message), 0);
            printf("Enter data for parent: ");
            gets(message);
            write(fd2[1], message, 1024);
            if(strcmp(message,"exit")==0) break;
        }
    }
}
```

Q3)1 way multiple time communication using tcp/ip

TCP SERVER

```
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#define PORT 8090
void main()
{
	int opt=1;
	int svrsock_fd,new_conn;
	char buffer[1024];
	struct sockaddr_in address;
	socklen_t addrlen=sizeof(struct sockaddr_in);
	svrsock_fd=socket(AF_INET,SOCK_STREAM,0);
	setsockopt(svrsock_fd,SOL_SOCKET,SO_REUSEADDR|SO_REUSEPORT,&opt,sizeof(opt));
	address.sin_family=AF_INET;
	address.sin_addr.s_addr=INADDR_ANY;
	address.sin_port=htons(PORT);
	bind(svrsock_fd,(struct sockaddr *)&address,addrlen);
	printf("WAITING FOR CLIENT\n");
	listen(svrsock_fd,3);
	new_conn=accept(svrsock_fd,(struct sockaddr *)&address,&addrlen);
	while(1)
	{
		memset(buffer,0,sizeof(buffer));
		read(new_conn,buffer,1024);
		printf("Received data from TCP/IP client:%s\n",buffer);
		if(strcmp(buffer,"exit")==0) break;
	}
}
```

TCP CLIENT

```
#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#define PORT 8090

void main()

{

        int clnsock_fd;

        char message[1024];

        struct sockaddr_in svraddr;

        socklen_t svraddrlen=sizeof(struct sockaddr_in);

        clnsock_fd=socket(AF_INET,SOCK_STREAM,0);

        svraddr.sin_family=AF_INET;

        svraddr.sin_addr.s_addr=inet_addr("127.0.0.1");

        svraddr.sin_port=htons(PORT);

        connect(clnsock_fd,(struct sockaddr*)&svraddr,svraddrlen);

        while(1)

        {

                memset(message,0,sizeof(message));

                printf("enter data for TCP/IP");

                gets(message);

                send(clnsock_fd,message,strlen(message),0);

                if(strcmp(message,"exit")==0)break;

        }

}
```

Q4)UDP CONNECTION

UDP SERVER

```
// server program for udp connection
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
        char buffer[100];
        char *message = "Hello Client";
        int listenfd, len;
        struct sockaddr_in servaddr, cliaddr;
        bzero(&servaddr, sizeof(servaddr));

        // Create a UDP Socket
        listenfd = socket(AF_INET, SOCK_DGRAM, 0);
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port = htons(PORT);
        servaddr.sin_family = AF_INET;

        // bind server address to socket descriptor
        bind(listenfd, (struct sockaddr*)&servaddr, sizeof(servaddr));

        //receive the datagram
        len = sizeof(cliaddr);
        int n = recvfrom(listenfd, buffer, sizeof(buffer), 0, (struct sockaddr*)&cliaddr,&len); //receive message from
server
        buffer[n] = '\0';
        puts(buffer);

        // send the response
        sendto(listenfd, message, MAXLINE, 0, (struct sockaddr*)&cliaddr, sizeof(cliaddr));
}
```

UDP CLIENT

```
// udp client driver program
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>

#define PORT 5000
#define MAXLINE 1000

// Driver code
int main()
{
        char buffer[100];
        char *message = "Hello Server";
        int sockfd, n;
        struct sockaddr_in servaddr;

        // clear servaddr
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

```
	servaddr.sin_port = htons(PORT);
	servaddr.sin_family = AF_INET;

	// create datagram socket
	sockfd = socket(AF_INET, SOCK_DGRAM, 0);

	// connect to server
	if(connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0)
	{
		printf("\n Error : Connect Failed \n");
		exit(0);
	}

	// request to send datagram
	// no need to specify server address in sendto
	// connect stores the peers IP and port
	sendto(sockfd, message, MAXLINE, 0, (struct sockaddr*)NULL, sizeof(servaddr));

	// waiting for response
	recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct sockaddr*)NULL, NULL);
	puts(buffer);

	// close the descriptor
	close(sockfd);
}
```