TERMWORK 1 (Pipes)

```
#include<unistd.h>
#include<stdio.h>
#include<sys/types.h>
#include<sys/wait.h>

int main()
{
	int fd[2]; char buffer[100];
	pipe(fd);
	pid_t p = fork();
	if(p > 0)
	{
		printf("Parent PID : %d\n", getpid());
		printf("Child PID  : %d\n", p);
		printf("[+]Passing \'Hello\' to child.\n\n");
		write(fd[1], "Hello", 5);
	}
	else
	{
		printf("Child PID  : %d\n", getpid());
		printf("Parent PID : %d\n", getppid());
		int n = read(fd[0], buffer, 100);
		printf("Child received the following data : %s\n", buffer);
	}
}
```

TERMWORK 1 (Queues)

Sending:

```
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

struct message{
        long type;
        char text[MAX];
}msg;

int main()
{
        key_t key = ftok("progfile", 65);
        int qid = msgget(key, 0666 | IPC_CREAT);
        msg.type = 1;
        printf("Enter the data to be written : ");
        fgets(msg.text, MAX, stdin);
        msgsnd(qid, &msg, sizeof(msg), 0);
        printf("Data sent is : %s \n", msg.text);
}
```

Receiving:

```
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

struct message{
        long type;
        char text[MAX];
}msg;

int main()
{
        key_t key=ftok("progfile", 65);
        int qid = msgget(key, 0666|IPC_CREAT);
        msgrcv(qid, &msg, sizeof(msg),1,0);
        printf("Data Received is : %s \n",msg.text);
        msgctl(qid, IPC_RMID, NULL);
}
```

TERMWORK 2

Server

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

#define PORT 4444
void main(){

  struct sockaddr_in server, client;
  socklen_t clilen = sizeof(client);
  char buffer[1024];

  int listenfd = socket(AF_INET, SOCK_STREAM, 0);
  printf("[+]Server socket created successfully.\n");

  bzero(&server, sizeof(server));
  server.sin_family = AF_INET;
  server.sin_port = htons(PORT);
  server.sin_addr.s_addr = htonl(INADDR_ANY);

  bind(listenfd, (struct sockaddr*)&server, sizeof(server));
  printf("[+]Socket bound to port number %d.\n", PORT);

  listen(listenfd, 5);
  printf("[+]Listening...\n");

  int connfd = accept(listenfd, (struct sockaddr*)&client, &clilen);
  printf("[+]Connection established.\n");
  strcpy(buffer, "Hello");
  send(connfd, buffer, strlen(buffer), 0);
  printf("[+]\'%s\' has been transmitted.\n", buffer);
  printf("[+]Closing the connection.\n");
}
```

Client

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

#define PORT 4444
void main(){

  int client = socket(AF_INET, SOCK_STREAM, 0);
  char buffer[1024];

  struct sockaddr_in server;
  printf("[+]Client Socket Created Successfully.\n");
  bzero(&server, sizeof(server));
  server.sin_family = AF_INET;
  server.sin_port = htons(PORT);
  server.sin_addr.s_addr = htonl(INADDR_ANY);

  connect(client, (struct sockaddr*)&server, sizeof(server));
  printf("[+]Connected to Server.\n");
  recv(client, buffer, 1024, 0);
  printf("[+]Data Received from Server : %s\n", buffer);
  printf("[+]Closing the connection.\n");
}
```

TERMWORK 3

```
#include<stdio.h>

struct node
{
   unsigned dist[10];
   unsigned to[10];
}rt[10];

int main()
{
   int cost[10][10], nodes, flag;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);
   printf("\nEnter the cost matrix :\n");
   for(int i=0;i<nodes;i++)
      for(int j=0;j<nodes;j++)
      {
         scanf("%d",&cost[i][j]);
         rt[i].dist[j]=cost[i][j];
         rt[i].to[j]=j;
      }
   do
   {
      flag = 0;
      for(int i=0;i<nodes;i++)
         for(int j=0;j<nodes;j++)
            for(int k=0;k<nodes;k++)
               if(rt[i].dist[j] > rt[i].dist[k] + rt[k].dist[j])
               {
                  rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];
                  rt[i].to[j] = k;
                  flag = 1;
               }
   }while(flag!=0);

   for(int i=0;i<nodes;i++)
   {
      printf("\n\nFor router %d:\n",i+1);
      for(int j=0;j<nodes;j++)
         printf("\t\nNode %d via %d Distance %d ",j+1,rt[i].to[j]+1,rt[i].dist[j]);
   }
   printf("\n\n");
}
```

TERMWORK 4

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc, char **argv){
   if (argc != 2) {
      printf("Usage: %s <port>\n", argv[0]);
      exit(0);
   }
   int port = atoi(argv[1]);
   struct sockaddr_in server, client;
   char buffer[1024];
   int sockfd = socket(AF_INET, SOCK_DGRAM, 0);
   printf("[+]Server Socket Created Successfully.\n");

   bzero(&server, sizeof(server));
   server.sin_family = AF_INET;
   server.sin_port = htons(port);
   server.sin_addr.s_addr = htonl(INADDR_ANY);

   int n = bind(sockfd, (struct sockaddr*)&server, sizeof(server));
   printf("[+]Socket bound to port number %s.\n", argv[1]);
   socklen_t clilen = sizeof(client);
   recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&client, &clilen);
   printf("[+]Data received : %s\n", buffer);

   bzero(buffer, 1024);
   strcpy(buffer, "Welcome to the UDP Server.");
   sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&client, clilen);
   printf("[+]Data sent     : %s\n", buffer);
}
```

Client

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

int main(int argc, char **argv){
    if (argc != 2) {
        printf("Usage: %s <port>\n", argv[0]);
        exit(0);
    }

    int port = atoi(argv[1]);

    struct sockaddr_in server;
    char buffer[1024];
    int sockfd = socket(AF_INET,SOCK_DGRAM,0);
    printf("[+]Client Socket Created Successfully.\n");

    bzero(&server, sizeof(server));
    server.sin_family = AF_INET;
    server.sin_port = htons(port);
    server.sin_addr.s_addr = htonl(INADDR_ANY);

    strcpy(buffer, "Hello World!");
    sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&server, sizeof(server));
    printf("[+]Data sent    : %s\n", buffer);

    socklen_t servlen = sizeof(server);
    bzero(buffer, 1024);
    recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&server, &servlen);

    printf("[+]Data received: %s\n", buffer);
}
```

NOTE

TERMWORK 5 CLIENT AND SERVER
SAME AS
TERMWORK 2 CLIENT AND SERVER