Part B Programs

1) Write a program for error detecting code using CRC-CCITT

```
#include <iostream>
#include <string.h>
using namespace std;
int arc(char *ip, char *op, char *poly, int mode)
{
    strcpy(op, ip);
    if (mode){
        for (int i=1; i<strlen(poly); i++)
            strcat(op, "0");
    }
    for (int i=0; i<strlen(ip); i++){
        if (op[i] == '1'){
            for (int j=0; j<strlen(poly); j++){
                if (op[i+j] == poly[j])
                    op[i+j] = '0';
                else
                    op[i+j] = '1';
            }
        }
    }
    for (int i=0; i<strlen(op); i++)
        if (op[i] == '1')
            return 0;
        return 1;
}
int main(){
    char ip[50], op[50], recv[50];
    char poly[] = "10001000000100001";
```

```
cout << "Enter the input message in binary"
cin >> ip;
crc(ip, op, poly, 1);
cout << "The transmitted message is:" << ip
    << op + strlen(ip) << endl;

cout << "Enter the received message in binary :" << endl;
cin >> recv;
if (crc(recv, op, poly, 0))
        cout << "No error in data" << endl;
else
    cout << "Error in data transmission has occurred" << endl;
    return 0;
}
```

Output

```
Enter the input message in binary
111101
The transmitted message is: 111101110100111010

Enter the received message in binary
111101
No error in data
```

2] Write a Program for congestion control using leaky bucket Algorithm

```
-> #include<bits/stdc++.h>
Using namespace std;
#define NOF_PACKETS 10
int rand(int a)
{
   int rn = (random() % 10) % a;
   return rn == 0 ? 1 : rn;
}
int main(){
   int packet_sz[NOF_PACKETS], i, clk,
   b_size, o_rate, p_sz, p_sz_rm = 0, p_time, op;
   for (i=0; i<NOF_PACKETS; i++){
         packet_sz[i] = rand(6) * 10;
   for (i=0; i<NOF_PACKETS; i++)
      printf("\n Packet[%d]:%d bytes\t",
                  i, packet_sz[i]);
   Printf("\n Enter the Output rate:");
   Scanf("%d", &o_rate);
   Printf("Enter the Bucket size:");
   Scanf("%d", &b_size);
   for (i=0; i<NOF_PACKETS; i++){
   if ((packet_size[i] + p_sz_rm) > b_size)
      if (packet_sz[i] > b_size)
         printf("\n\n Incoming Packet size
               (%d bytes) is Greater than
                  bucket capacity
```

```
else
{   P_sz_rm += Packet_sz[i];
    Printf("\n\n Incomming Packet size: %d",
                Packet_size %d ", Packet_sz[i]);
    Printf("\n Bytes remaining to Transmit %d", P_sz_rm);
    P_time = rand(4)*10;
    Printf("\n time left for transmission: %d units", P_time);
    for(clk=10; clk<= P_time; clk+=10){
        Sleep(1);
        if(P_sz_rm){
            if(P_sz_rm <= o_rate) op = P_sz_rm = 0
            else
            Printf("\n Packet of size %d transmitted ", op);
            Printf("Bytes remaining to transmit: %d", P_sm_rm);
        }
        else{
            Printf("\n Time left for transmission: %d unit",
                        P_time - clk);
            Printf("\n NO Packets to transmit)";
        }
    }
}
}
}
```

Output :

Packet[0] : 30 byte
Packet[1] : 10 byte
Packet[2] : 10 bytes
Packet[3] : 30 bytes
Packets[4] : 30 byte

Enter Output rate : 100
Enter bucket size : 50

Incomming packet size : 30
Bytes remaining to transmit : 30
Time left for transmission : 20 units
Packet of size 30 transmitted .. Bytes remaining to transmit :
Time left for transmission : 0 units
No packet to transmit

Incoming Packets size : 10
Bytes remaining to transmit : 100
Time left for transmission : 30
Packet of size 10 transmitted .. Byte remaining to transmit : 0
Time left for transmission : 10
No packet to transmit :
Time left for transmission : 0 units
No packet to transmit :

Incoming packet size : 10
Byte remaining to transmit : 10
Time left for transmission : 10 units
Packet of size 10 transmitted... Byte remaining to transmit : 0

Incoming Packet size : 30
Byte remaining to transmit : 30
Time left for transmission : 30 units
Packet of size 30 transmitted... Byte remaining to transmit : 0
Time left for transmission : 10 units
No Packets to transmit
Time left for transmission : 0 units
No Packets to transmit.

3] Using TCP/IP sockets, write a client-server program to make client sending the filename & the server to send back the contents of requested file if present

Client side :

```
#include <unistd>
int main() {
int soc, n;
char buffer[1024], fname[50];
struct sockaddr_in addr;
soc = socket(PF_INET, SOCK_stream, 0);
addr.sin_family = AF_INET;
addr.sin_port = htons(7891);
addr.sin_addr.s_addr = inet_addr("127.0.0.1");
while (connect(soc, (struct sockaddr *) &addr, sizeof(addr)
Printf("\n Enter file name");
Scanf("Enter file name");
Scanf("%s", fname);
Send(soc, fname, size of (fname), 0);
Printf("\n Received response \n");
while((n = recv(soc, buffer, size of (buffer), 0)) > 0)
        Printf("%s", buffer);
        return 0;
}
```

Server side:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/socket.h>
int main() {
    int welcome, new_soc, fd, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;
    welcome = socket(PF_INET, SOCK_STREAM, 0);
    addr.sin_family = AF_INET;
    addr.sin_port = htons(7891);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    bind(welcome, (struct sockaddr) &addr, sizeof(addr));
    printf("\n Server is online");
    listen(welcome, 5);
    new_sock = accept(welcome, NULL, NULL);
    recv(new_soc, fname, 50, 0);
    Printf("\n Requesting for file : %s\n", fname);
    fd = open(fname, O_RDONLY);
    if (fd < 0)
        send(new_soc, "\n File not found \n", 50, 0);
    else
    while ((n = read(fd, buffer, sizeof(buffer)) > 0
        send(new_soc, buffer, &n, 0);
    printf("\n Request sent \n")
    close(fd)
    return 0;
}
```

Output

Server is Online
Requesting for file : test.txt
Request Send
Client is connected to server
Enter file name : test.txd
Received response
Hello Helloworld.

4] Using UDP sockets, write a client - server Program to make client sending the filename & the server to send back contents of the requested file if present.

Server Program:

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define PORT 5000
#define MAXLINE 1000
int main()
{
    char buffer[100];
    char message = "Hello client";
int listenfd, len;
struct sock_in servaddr, cliaddr;
bzero(&servaddr, sizeof(servaddr));
listenfd = socket(AF_INET, SOCK_DGRAM, 0);
servaddr.sin_port = htons(PORT);
bind(listenfd, (struct sockaddr*)&servaddr, sizeof(addr)
len = sizeof(cliaddr)
int n = recvfrom(listenfd, buffer, sizeof(buffer), 0,
        (struct sockaddr*)&cliaddr, &len);
buffer[n] = '\0';
puts(buffer)
sendto(listenfd, message, MAXLINE, 0, (struct sockaddr*)
    &clintaddr, sizeof(cliaddr));
}
```

Client driven Program

```
#include <stdio.h>
#include <strings.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <unistd.h>
#include <stdlib.h>
#define PORT 5000
#define MAXLINE 1000
int main() {
    char buffer[100];
    char *message = "Hello server";
    int sockfd, n;
```

```
Struct sockaddr_in servaddr;
bzero (& servaddr, sizeof(servaddr));
Servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
Servaddr.sin_port = htons(PORT)
Sockfd = socket(AF_INET, SOCK_DGRAM, 0)
if (connect (sock_fd, (struct sockaddr*) & servaddr.
                                    sizeof(servaddr))
    Printf("In Error: Connect Failed\n");
    exit(0);
}
Send to (sock fd, message, MAX(INET, 0, (struct
                                    Sockaddr*) NULL)
recfrom(sock_fd, buffer, sizeof(buffer), 0, (struct
                    sockaddr*) NULL, NULL);
    puts (buffer
    Close (sock_fd);
}
```

Output

Server Output

Server is online

HELLO SERVER

Client Output

Hello Client.