PAGE NO :
DATE :

## PART-B

01. Write a program for error detecting code using CRC - CCIT (16 bits)

```
#include <iostream>
#include <string.h>
using namespace std;
int crc (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 input msg in binary" << endl;
  cin >> ip;
```

```
crc(ip, op, poly, 1);
cout<< "Transmitted msg is: "<< ip << op+strlen(ip)<< endl;
cout<< "Enter received msg in binary"<<endl;
cin >> recv;
if (crc(recv, op, poly, 0))
   cout << "No error in data"<<endl;
else
   cout << "Error has occurred"<<endl;
return 0;
}
```

Output:

```
Enter input msg in binary
111101
Transmitted msg is: 111101101011100111010
Enter received msg in binary
111101
No error in data
```

02. Write a program for congestion control using leaky bucket algorithm

```
#include<iostream>
#include<string.h>
#using namespace std;
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define no_of_packets 10
int rand(int a){
   int rn = (random()%10)%a;
   return rn==0?1:rn;
}
```

PAGE NO :
DATE :

```
int main(){
 int packets[no_of_packets], i, clk, b_size, o_rate,
    p_sz_rm=0, p_sz, p_time, op;
 for(i=0; i<no_of_packets; ++i)
    packet_sz[i]= rand(6)*10;
 }
 for(i=0; i<no_of_packets; ++i)
    printf("\n packet[%d]: %d bytes\t", i, packet_sz[i]);
 printf("\n Enter output rate:");
 scanf("%d", &o_rate);
 printf("Enter bucket size:");
 scanf("%d", &b_size);
 for(i=0; i<no_of_packets; ++i){
    if((packet_sz[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 (%d bytes)-
              Packet Rejected", packet_sz[i], b_size);
      else
        printf("\n\n Bucket capacity exceeded - Rejected");
    }
    else{
      p_sz_rm+=packet_sz[i];
      printf("\n\n Incoming 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, p_sz_rm=0;
          else                     op=o_rate, p_sz_rm-=o_rate;
          printf("\n Packet of size %d transmitted", op);
```

```
            printf("...Bytes remaining to transmit: %d", p-sz);
        }
        else {
            printf("\n Time left for transmission: %d unit",
                    p->time_clk);
            printf("\n No packets to transmit!");
        }
    }
  }
 }
}
```

Output:

packet[0]: 30 bytes
packet[1]: 10 bytes
packet[2]: 10 bytes
packet[3]: 50 bytes
packet[4]: 30 bytes

Enter output rate: 100
Enter bucket size: 50

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

Incoming packet size: 10
Bytes remaining to transmit: 10
Time left for transmission: 30

PAGE NO :
DATE :

Packet of size 10 transmitted.. Bytes remaining to transmit:0
Time left for transmission: 10 units
No packet to transmit!
Time left for transmission: 0 units
No packets to transmit!

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

Incoming packet size: 50
Bytes remaining to transmit: 50
Time left for transmission: 10 units
Packet of size 50 transmitted.. Bytes remaining to transmit:0

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

03. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Client side:

```
#include <unistd.h>
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 Client is connected to server");
printf("\n Enter file name: ");
scanf("%s", fname);
send(soc, fname, sizeof(fname), 0);
printf("\n Received response\n");
while((n = recv(soc, buffer, sizeof(buffer), 0)) > 0)
  printf("%s", buffer);
return 0;
}
```

Server side:

```
#include <stdio.h>
#include <arpa/inet.h>
#include <fcntl.h>
#include <unistd.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));
```

PAGE NO :
DATE :

```
printf("\n Server is online");
listen(welcome, 5);
new_soc = 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", 15, 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;
}
```

No Open

4. Using UDP sockets, write a client-server program to make client sending the file name and 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>
#include<netinet/in.h>
#define PORT 5000
#define MAXLINE 1000
int main(){
  char buffer[100];
  char *message = "Hello Client";
```

```
int listenfd, len;
struct sockaddr_in servaddr, cliaddr;
bzero(&servaddr, sizeof(servaddr));
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(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
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 *) &cliaddr, sizeof(cliaddr));
}
```

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
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);
servaddr.sin_family = AF_INET;
sockfd = socket(AF_INET, SOCK_DGRAM, 0);
if(connect(sock_fd, (struct sockaddr*) &servaddr, sizeof
                                        (servaddr))<0)
{
  printf("\n Error: Connect Failed\n");
  exit(0);
}
sendto(sockfd, message, MAX(INF,0, (struct sockaddr*)
        NULL, sizeof(servaddr));
recvfrom(sockfd, buffer, sizeof(buffer), 0, (struct
        sockaddr*) NULL, NULL);
puts(buffer);
close(sockfd);
}
```