DATE: PAGE:

PART - B

Q1. 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
}
```

Scanned with CamScanner

DATE: PAGE:

```
int main ()
{
    char ip[50], op[50], recv[50];
    char poly[] = "10001000000100001";
    cout << "Enter ip msg in binary" << endl;
    cin>>ip;
     crc(ip,op,poly,1);
     cout << "Transmitted msg is" << ip<<op+strlen(ip)
                                              << endl;

    cout<<"Enter recieved msg in binary"<<endl;
    cin >>recv;
    if(crc(recv,op,poly,0))
      cout<<"No error"<<endl;
    else
      cout<<"Error in transmission"<<endl;
    return 0;
}
```

op

```
Enter input message in binary
 1111101
The transmitted message is : 111110110101111011
                                              010
Enter recieved message
1111101
No error in data
```

2)

```
Enter input msg in binary : 1111101
The transmitted msg is : 111110110101111100111010
Enter recieved message in binary: 1110
Error in data transmisson occured
```

2) 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 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_rm=0, p_sz, p_time, op;
  for (i=0; i<NOF_PACKETS; ++i)
      packet_sz[i] = rand(6)* 10;
  for(i=0; i<NOF_PACKETS; ++i)
      printf("packet [%d]: %d bytes :", packet_sz[i]);
  printf(" Enter op rate");
  scanf("%d", &o_rate);
  printf(" Enter Bucket size ");
  scanf("%d", &b_size);
  for(i= 0; i<NOF_PACKETS; ++i) {
      if (packet_sz[i] + p_sz_rm) > b_size)
          if (packet_sz[i] > b_size)
              printf("Incoming packet size (%d bytes) is
              greater than bucket capacity (%d bytes)-
              PACKET REJECTED", packet_sz[i], b_size[i]);
else
    printf ("Bucket capacity exceeded.
     PACKETS REJECTED");
else
{
    p_sz_rm += packet_sz[i];
    printf ("Incoming packet size: %d", packet_sz[i]);
    printf(" Bytes remaining : %d", p_sz_rm);
    p_time = rand(u)*10
    printf("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 ("Packet of size %d Transmitted", op);
    printf (" ---Bytes Remaining to Transmit:
                %d", p_sz_rm); }
    else {
          ~~printf("Time left to transmit: %d units,
                     p_time - clk);~~
          printf(" No packets to transmit");
     } }
  } }
}
```

O/P
```
packet[0]: 30 bytes
packet[1]: 10 bytes
packet[2]: 10 bytes
```

DATE: PAGE:

```
Enter Output rate:100
Enter Bucket size:50

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

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

Incoming packet size: 10
Bytes remaining to transmit: 10
Time left for transmission: 10 units
Packets of size 10 transmitted -- Bytes
                    remaining to transmit: 0
```

DATE:

PAGE:

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

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
   while (connect (soc, (struct sockaddr *) &
           addr, sizeof (addr)));
   printf ("Client is connected to server");
   printf ("Enter file name");
   scanf ("%s", fname);
   send (soc, fname, sizeof (fname), 0);
   printf (" Recieved response");
   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>
```

DATE: PAGE:

```
int main(){
 int welcome, new_soc, fd, n;
 char buffer [1024], fname [50];
 struct sockaddr_in addr;
 welcome = socket (PF_INET
 addr.sin_family = AF_NET
 addr.sin_port = htons(7891);
 addr.sin_addr.s_addr = inet_addr("127.0.0.1");
 bind(welcome, (struct sockaddr *)&addr, sizeof(addr));
 printf ("Server Online)
 listen (welone, 5);
 new_soc= accept (welcome, NULL, NULL);
 recv (new_soc, fname, 50, 0);
 printf (" Requesting for file: %s", fname);
 fd= open( fname, O_RDONLY);
 if (fd<0)
     send (new_soc, "file not found", 15, 0);
 else
   while ((n = read(fd, buffer, sizeof(buffer)))>0)
      send(new_soc, buffer, n, 0);
  printf(" Request sent");
  close(fd);
  return 0;
}
```

DATE: PAGE:

4) Using UDP sockets, write a client server program to make client sending filename and the server to send back the contents of the requested file if present.

```
#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 *msg = "Hello Client";
   int listenfd, len;
   struct sockaddr_in servaddr, cliaddr;
    bzero( &servaddr, sizeof(servaddr));
    listenfd= socket(AF_INET, SOCK_DGRAM, 0);
     serveraddr.sin.addr.s_addr= htonl(INADDRAN
    servaddr.sin_port = htons (PORT);
    serveaddr.sin_family = AF_INET;
    bind(listenfd, (struct sockaddr*)&
     servaddr, sizeof( servaddr));
    len= sizeof( cliaddr);
     int n= recvfrom (listenfd, buffer, sizeof(buffe
            0, (struct sockaddr*)& cliaddr, &len);
     buffer [n]='\0';
     puts (buffer);
     sendto (listenfd, msg, MAXLINE, 0,
           (struct sockaddr*) &cliaddr, sizeof(cliaddr
  }
```

DATE: PAGE:

```
// 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 *msg = "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(sockfd, (struct sockaddr *) &servaddr,
          sizeof(servaddr)) < 0)
     {
         printf("Error: Connect Failed");
         exit(0);
     }
    sendto(sockfd, msg, MAXLINE, 0, (struct sockaddr*)
    NULL, sizeof(servaddr));
    recvfrom(sockfd, buffer, sizeof(buffer), 0,
         (struct sockaddr*) NULL, NULL);
     puts(buffer);
     close(sockfd);
}
```

[illegible] 30/12