PAGE NO :
DATE : 22-12-24

Part B

1. Write a program for error detecting code using CRC-CCITT (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;
}
```

PAGE NO :
DATE :

```
int main ()
{
    char ip[50], op[50], recv[50];
    char poly[] = "10001000000100001";

    cout << "Enter the input message in binary" <<
                                        endl;
    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
                occured" << endl;
    return 0;
}
```

//Output

Enter the input message in binary

1111101

The transmitted message is : 1111101101011111001110

Enter the received message in binary

1111101

No error in data

PAGE NO :
DATE : 23-11-24

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("\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);
```

PAGE NO :
DATE :

```
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\nIncomming packet size
      (%d bytes) is Greater than buck
      capacity (%d bytes) - Packet rejec
      packet_sz[i], b_size);
    else
      printf("\n\nBucket capacity exceeded
          Packets rejected!!");
  else
  {
    p_sz_rm += packet_sz[i];
    printf("\n\nIncomming Packet size %d
                packet_sz[i]);
    printf("\nBytes remaining to Transmit: %d
                p_sz_rm);
    p_time = rand(4) * 10;
    printf("\nTime 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 = p_sz_rm - o_rate;
        printf("\nPacket of size %d Transmitted
                op);
```

PAGE NO :
DATE :

```
                printf("... Bytes Remaining to Transmit:
                        %d", p->sz - rm);
            }
            else
            {
                printf("\n Time left for transmission: %d
                        units", 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 the output rate: 100
Enter the 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!!

PAGE NO :
DATE :

Incoming packet size: 10
Bytes remaing to Transmt: 10
Time left for trasmission: 30
Packet of size 10 Transmitted -- Bytes remaining to Trasmt: 0

Time left for trasmission: 10 units
No packets to trasmit!!
Time left for trasmission: 0 units
No packets to trasmit!!

Incoming packet size: 10
Bytes remaing to Trasmt: 10
Time left for trasmission: 10 units
Packet of size 10 Transmitted -- Bytes remaing to Trasmt: 0

Incoming Pacet size: 50
Bytes remaing to trasmt: 50
Time left for trasmission: 10 units
Packet of size 50 Trasmitted -- Bytes remaing to Trasmit: 0

Incoming Packet size: 30
Bytes remaing to Trasmt: 30
Time left for trasmission: 30 units
Packet of size 30 trasmitted -- Bytes remaing to trasmit: 0
Time left for trasmission: 10 units
No packets to trasmt!!
Time left for trasmission: 0 units
No packets to trasmit!!

PAGE NO :
DATE : 21-12-24

3 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<stdio.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))!=0);
    printf("\nClient is connected to server");
    printf("\nEnter file name:");
    scanf("%s", fname);

    send(soc, fname, sizeof(fname), 0);
    printf("\nReceived response\n");

    while((n=recv(soc, buffer, sizeof(buffer),0))>0)
        printf("%s", buffer);
    return 0;
}
```

PAGE NO :
DATE :

» server.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<sys/stat.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 Sever 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);
```

PAGE NO :
DATE :

```
else
    while((n=read(fd,buffer,sizeof(buffer)))
                                    >0)
        send(new_soc,buffer,n,0);

printf("\nRequest sent\n");
close(fd);

return 0;

}
```

26/12/24.

NO O/P

//output

Server is online
Requesting for file: test.txt
Request sent

client is connected to server
Enter file name: test.txt
Received response
Hello, World

PAGE NO :
DATE : 23-12-24

4. Using UDP 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

// 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, bn;
  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));
```

PAGE NO :
DATE :

~~int n~~

len = sizeof(cliaddr)

int n = recvfrom(listenfd, buffer, sizeof(buff),
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<string.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;
```

PAGE NO :
DATE :

```
struct sockaddr_in servaddr;

bzero(&servaddr, sizeof(servaddr));
servaddr.sin_addr.s_addr = inet_addr(
servaddr.sin_port = htons(PORT);
servaddr.sin_~~port = ht~~ family = AF_INET;

sockfd = socket(AF_INET, SOCK_DGRAM,

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

sendto(sockfd, message, MAXLINE, 0,
        (struct sockaddr*) NULL,
         sizeof(servaddr));

recvfrom(sockfd, buffer, sizeof(buffer), 0,
          (struct sockaddr*) NULL, NULL
puts(buffer);

close(sockfd);
}
```

26/12/24

O/P.

No

PAGE NO :
DATE :

// Sever Output
sever is Online
Hello Sever

// client Output
Hello client