DATE: PAGE:

PART-B

1) Write a program for error detection code using CRC-CCITT (16-bit)

```
# include <iostream.h>
# include <string.h>

using namespace std

int crc( char *ip, char *op, char *poly, int mod
{
    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
  char

  cout
  cin
  cr
  co

}
```

OUT

Ente
111
The
Ent
111

N

DATE: PAGE:

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

    cout << "Enter 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: 111110110101110011011
Enter the received message in binary
1111101

No error in data
```

DATE:

PAGE

2) write a program for congestion control wi
leaky bucket algorithm

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

int main()
{
    int packetsz [no_of_packet], i, clk, bsz, o, rt
      p_sz_rm = 0, p_sz, p_time, op,

    for (i = 0, i < no_of_packet ; ++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 < of_packets ; ++i)
    {

        if (packet_sz[i] > b_size)
        {

        printf ("\n
```

DATE: PAGE:

```
(%d bytes) is greater than bucket capacity (%d bytes)
 - Packet rejected", packet_sz[i], b_size);
else
   printf("\n\n Bucket capacity exceeded, packet
   rejected");
else
   {
      p_sz_trm = packet_sz[i];
      printf("\n\n Incoming packet size %d",
                                     packet_sz[i]);
      printf("\n Bytes remaining to transmit %d",
                                     p_sz_trm);
      p_time = rand(4) * 10;
      printf("\n Time left for transmission %d",
                                     p_time);
   for (clk = 10; clk <= p_time; clk += 10)
   {
      sleep(1);
      if (p_sz_trm)
      {
         if (p_sz_trm <= o_rate)
            op = p_sz_trm, p_sz_trm = 0;
      }
      else
            op = o_rate, p_sz_trm -= o_rate;
      printf("\n Packet of size %d transmitted", op);
      printf("Bytes remaining to transmit %d",
                                     p_sz_trm);
      }
      else
      {
      printf("\n time left for transmission %d",
                                     p_time - clk);
```

```
                printf("\n no of 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 : 80

~~Exe~~ Incoming packet size : 30
Bytes remaining to transmit : 30

Time left for transmission : 70 units
Packet of size 30 transmitted -- bytes remaining to trans

Time left for transmission : 0 units
No packets to transmit

Incoming packet size : 10
Bytes remaining to transmit : 10
Time left for transmission : 30
Packet of size 10 transmitted - Bytes remaining to
                                              transmit : 0
```

```
Time left
No packets
Time left
No packets

Incoming
Bytes re
Time le
Packet

Incomi
Bytes le
Time le
Packet

~~Ex~~ In
Bytes
Time
Packe

Time
No
Time
No
```

Time left for transmission : 10 unit
No packets to transmit
Time left for transmission : 0 unit
No packets to transmit

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

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

~~[illegible]~~ 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 unit
No packets to transmit
Time left for transmission = 0 unit
No packets to transmit.

DATE: PAGE:

1) Using TCP/IP socket, write a program to make 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 sox, n;
    char buffer[1024], fname[50];
    struct sockaddr_in addr;

    sox = socket (AF_INET, SOCK_STREAM, 0);
    addr. sin_family = AF_INET
    addr. sin_port = htons (4841);
    addr. sin_addr. s_addr = inet_addr ("127.0.0.1")

    while( connect (sox, (struct sockaddr *) &addr,
            sizeof (addr)));
    printf("\n client is connected to server ");
    printf("\n Enter file ~~name~~ name :");
    scanf("%s", fname).

    send (sox, fname, sizeof (fname) 0),
    printf("\n Received response \n"),

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

→ Serve

```
# includ
# includ
# inclu
# includ

int m
{
```

→ <u>Server side</u>

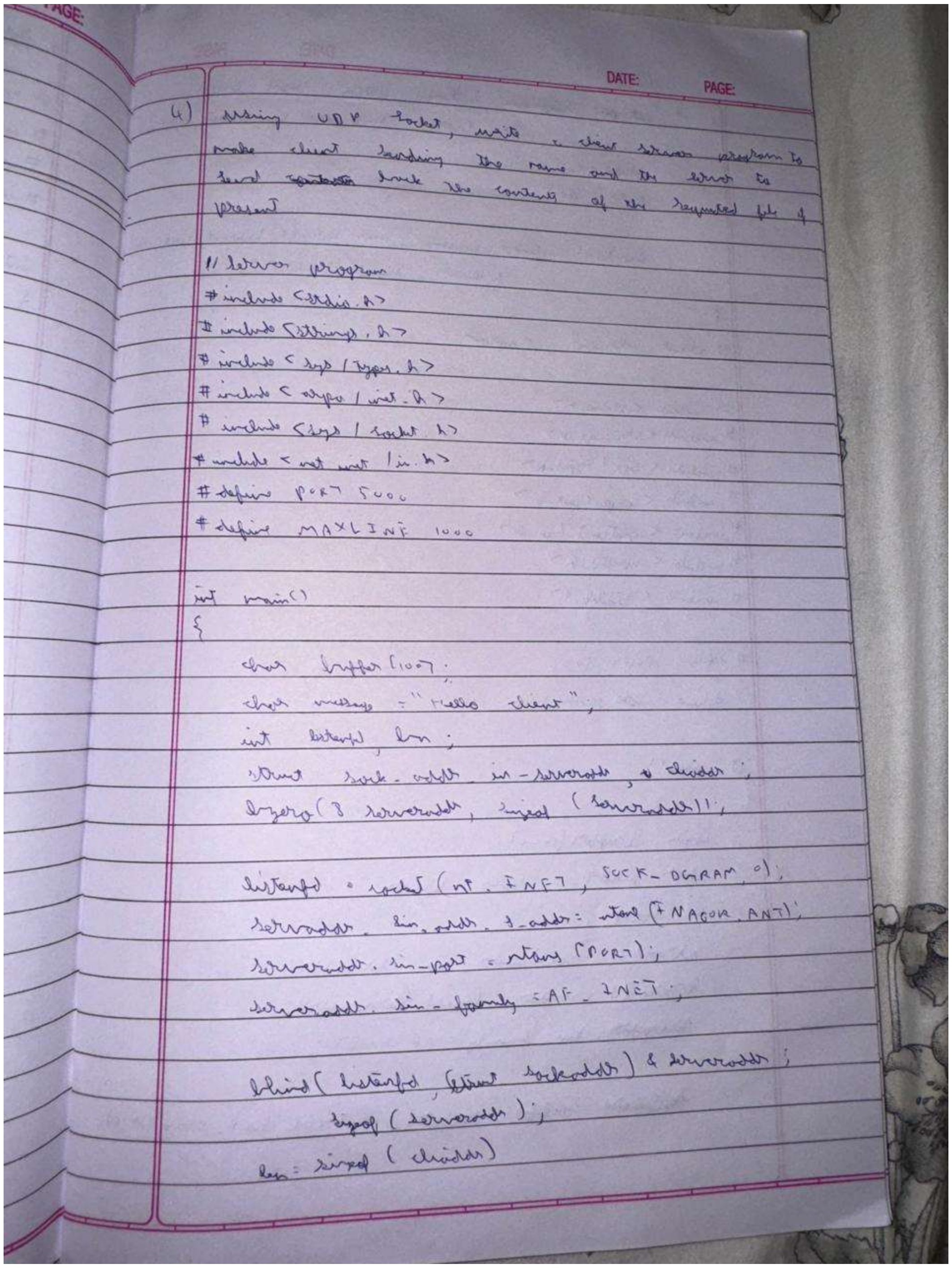
DATE: PAGE:

```
#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));
    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);
```

DATE: PAGE:

```
        printf ("\n Request sent \n");
        close (fd);
    return 0;
}
```

```
// ~~[illegible]~~ output
server is online
~~Hello~~ Requesting for file : test.text
Request sent

client is connected to server
Enter file name: test.txt
Received response
Hello world
```



DATE: PAGE:

4) Using UDP socket, write a client server program to make client sending the name and the server to send ~~[illegible]~~ 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, len;
    struct sock_addr_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)
```

DATE: PAGE:

```
    int n = recvfrom ( listenfd, buffer, sizeof (buffer),
                 0, (struct sockaddr ), &lenaddr,
    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 <netinet/in.h>
# include <unistd.h>
# include <stdlib.h>

# define PORT 5000
# define ~~[illegible]~~ MAXLINE 1000

int main()
{
    char buffer[1000];
    char message = "Hello server";
    int sockfd, n;
    struct sockaddr_in servaddr;
    bzero (& servaddr, sizeof (servaddr));
    servaddr.sin_family = AF_INET;

    ~~[illegible]~~ sockfd = socket (AF_INET, SOCK_DGRAM,
if (connect (sock_fd, (struct sockaddr) & servaddr,
                    sizeof (servaddr)) < 0)
{
    printf("\n Error : connection failed");
    exit(0);
}
sendto ( sockfd, message, MAX( INE, 0, (struct sockaddr),
                    NULL, sizeof( serveraddr));

recvfrom( sockfd, buffer, sizeof ( buffer), 0,
            (struct sockaddr), NULL, NULL);
puts (buffer);
close (sockfd);
}
```

```
// Server output
Server is online
Hello Server

//client output
Hello client
```