Subject :

Software :

Hardware :

Branch :

Semester :

Page No.

Prog No. 3

OBLEM STATEMENT Write a Program to compute CRC code for the Polynomials CRC - 12, CRC - 16 and CRC CCIP in network.

GORITHM & CODE :

To compute the CRC for different polynomials like CRC-12, CRC-16 and CRC CCITT in a network communication content we need to ~~β~~ define the respective polynomials and implement algorithm to calculate the CRC.

Algorithm for CRC computation :-

1. Define the polynomial
2. Initialize the CRC.
3. Process the Data
4. Final CRC.

C Program for CRC Calculation.

```
#include <stdio.h>
#include <stdint.h>
#define CRC-12 POLY 0x 80F
#define CRC_12 - INIT 0xFFF
#define CRC -12 _ x OR OUT 0x0
#define CRC-16 - POLX
#define CRC-16 - INIT 0x FFFF
#define CRC-16 - xOROUT 0X 0000
#define CRC-16 -
#define CRC - CCITT - POLY 0x 11021
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :			Software :	
			Hardware :	
Branch :		Semester :	Page No.	Prog No.

OBLEM STATEMENT

GORITHM & CODE :

```
#define CRC_CCITT_INIT  0x ffff
#define CRC_CCITT_ XOROUT  0x0000

unit 16_t compute_crc (uint8_t* data, size_t
length, uint16_ t poly, uint 16_t init_value, uint 16_t
xor_out){
      uint 16_t crc = init_value;
      for (size_t i=0; i< length ; i++){
            crc ^= (data [i]<< 8);
            for (int j=0; j<8; j++){
                  if (crc & 0x8000){
                        crc = (crc << 1)^ poly;
                  }else{
                        crc <<=1;
                  }
                  crc &= 0xffff ;
            }
      }
      crc ^= xor_out;
      return crc & 0x ffff ;
  }
  unit 16_t crc12 (uint8_t *data, size_t length){
        return compute_crc (data, length , CRC_12 _POLY,
CRC_12_INIT , CRC_12 _XOROOT);
}
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student
Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No.	Prog No.

OBLEM STATEMENT

LGORITHM & CODE :

```
//function to compute CRC-16
uint16_t crc16 (uint8_t *data, size_t length){
        return compute_crc (data, length, CRC_12_
POLY, CRC_16_INIT, CRC_16_XOROUT);
}

//Function to compute CRC-16
unit 16_t crc_ccitt (uint8_t *data, size_t length){
        return compute_crc (data, length, CRC_CCITT_POLY
CRC_CCITT_INIT, CRC_CCITT_XOROUT);
}

int main (){
    //Test data (bytearray
    unit8_t data[] = {0x31, 0x32, 0x33, 0x34);
    size_t length = sizeof (data)/sizeof (data[0]);

//Compute CRCs for the given data
unit16_t crc12_result = crc12(data, length);
unit16_t crc16_result = crc16 (data, length);
unit 16_t crc_ccitt_result = crc_ccitt (data, length);
//Display the results
Print ("CRC-12 : %03X\n", crc12_result);
Print ("CRC-16 : %04X\n", crc16_result);
Print ("CRC-CCITT : %04X\n", crc_ccitt_result);
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

OBLEM STATEMENT

GORITHM & CODE :

```
return 0;
}
```

Output :

If the input data is "1234" ,the output will be:

```
CRC-12 : 6F7
CRC-16 : 29B1
CRC - CCITT : 31C5
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

BLEM STATEMENT Develop a Simple data link layer that Performs the Flow control using the sliding window protocol, and loss recovery using the Go-Back N mechanism.

GORITHM & CODE:

To develope a Simple Data link layer (DLL) in C that performs flow control using the sliding window Protocol and less recovery using the Go-Back N (GBN) mechanism, we need to implement the following components:

1. Sliding ~~the~~ Window Protocol: This is used for flow control, where the sender can send multiple frames before needing an acknowledgment, but it has to maintain a window of acknowledged frames.

2. Go-Back N (GBN): This is a mechanism where the receiver acknowledges frames, but the sender can only send up to N frames without receiving an acknowledgment. If a frame is lost or corrupt, all subsequent frames are retransmitted.

Basic Concepts:

- Sender side: keeps track of a window of sent frames and waits for acknowledgments.

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE : Signature of Faculty Signature of Student

Date : Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No.	Prog No.

OBLEM STATEMENT

GORITHM & CODE :

• Receiver Side: Receives frames in Order and Sends back an acknowledgement.

• Sequence Numbering: Frames are assigned sequence numbers, and acknowledgment are expected to be received with the corresponding sequence number.

Structure:

1. Sender Window: The Sender has a "window" of frames it is allowed to send. The size of this window is N.

2. Receiver Window: The receiver expects frames in Order, but can buffer frames out of order, as long as they are within the window.

Simple code Implementation in C:

```
#include <Stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <String.h>
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :

Software :

Hardware :

Branch :

Semester :

Page No.

Prog No.

BLEM STATEMENT

GORITHM & CODE :

```
# include <Pthread.h>
# include <time.h>

# define MAX_FRAME 10
# define TIMEOUT 2
# define WINDOW_SIZE4

typedef Struct {
    int Seq_num;
    Char data [100];
  } frame;

frame Sender_buffer [MAX FRAME];
Frame receiver_buffer [MAX_FRAME];

int Sender_base =0;
int next_seq_num =0;
int receiver expected_Seq_num = 0;
int ack_received [MAX_FRAME] ={0};
int is_running = 1;
Pthread_mutex_t lock;
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :

Software :

Hardware :

Branch :

Semester :

Page No.

Prog No.

OBLEM STATEMENT

GORITHM & CODE :

```
void * acknowledge_thread(void * arg){
    while (is_running){
        sleep(1);
        Pthread_mutex_lock(&lock);
        if (sender_base < next_seq_num){
            Printf("Receiver side; checking frame
%d\n", receiver_expected_seq_num);
            receiver_expected_seq_num =
(receiver_expected_seq_num +1) % MAX_FRAME;

        ack_received [Frame.seq_num]= 1;
        }else{
            Printf("Receiver: Out of order frame%d,
 expecting frame %d\n", frame.seq_num, receiver_
expected_seq_num);
        }
    }
    Pthread_mutex_unlock (&lock);
  }
  return NULL;
}
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :

Software :

Hardware :

Branch :

Semester :

Page No.

Prog No.

BLEM STATEMENT

GORITHM & CODE :

```
//Sender thread that simulates Sending frames.
void * Seder_thread (void *arg){
    While (is_running){
        Pthread_mutex_lock (& lock);

        if (next_seq_num < sender_base + WINDOW_
SIZE && net_seq_num <MAX_FRAMES){
    frame frame;
    frame.seq_num = next_seq _num;
     snprintf (frame.data, sizeof (frame.data),
"frame %d", next_seq_num);
      Send_frame( frame);
      next_seq_num ++;
}
for (int i = sender_base ; i<next_seq_num; i++){
    if (ack_received [i]){
        Sender_base = (sender _base +1 )
%MAX_FRAME ;
        Printf ("Sender: Acknowledgment received for
frame %d , Sliding window \n", i);
    }   break;
}
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No.	Prog No.

BLEM STATEMENT

ORITHM & CODE :

```
    Pthread_mutex_unlock (&lock);
    sleep (TIMEOUT);
  }
  return NULL;
}

int main () {
   Srand (time (NULL));
     pthread_t sender, receiver;
      for (int i = 0; i < MAX_FRAME; i++){
          snprintf (Sender_buffer [i].data, sizeof
 (sender_buffer [i], data), "frame %d", i);
 }
 for (int i = 0; i < MAX_FRAME ; i++){
        snprintf (receiver_buffer [i] .data, sizeof(
 receiver _buffer [i] .data), "frame %d", i);
 }
 Pthread_mutex_init (& lock , NULL);
 Pthread_create (&sender, NULL , Sender_thread , NULL);
 Pthread_create (&receiver, NULL, acknowledge_thread, NULL);
 Sleep (10);
 is_running = 0;
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :
		Hardware :
Branch :	Semester :	Page No. | Prog No.

ROBLEM STATEMENT

LGORITHM & CODE :

```
Pthread_join (sender, NULL);
Pthread_join (receiver, NULL);

Pthread_mutex_destroy (&lock);
Printf ("sender has finished sending.\n");

return 0;
}
```

o/p:

```
Sending frame : 0
Receiver side: Checking frame 0
Receiver : Out of order frame 3, expecting frame 0
Receiver side: checking frame 0.
Receiver : Out of order frame 5, expecting frame 0
Sending frame: 1
Receiver side: checking frame 0
Receiver : Out of order frame 1, expecting frame 0
Sending frame : 2
Receiver side: checking frame 0.
Receiver: Out of order frame 6, expecting frame 0.
Receiver side: checking frame 0.
Receiver : out of order frame 2 , expecting frame 0
Sending frame 3
Receiver side : checking frame
Receiver: Out of order frame 9, expecting frame 0.
```

INPUT GIVEN

OUTPUT OBTAINED Sender has finished sending.

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :