C.N Lab 1

Title:
Implementation of Cyclic Redundancy Check (CRC)
for error detection.

Lohith Kandula, AP20110010161, lohith_ranganadhareddy@srmap.edu.in ,
CSE C
Submitted on: 28 August 2022

Objective:

Data security is a significant factor while it is being transmitted through the network. When the message is sent from sender to receiver it can be corrupted by a noisy channel.

So, we need an error detection algorithm to check if the message is corrupted or not. There are many algorithms for error detection but as of now, we implement Cyclic Redundancy Check(CRC) to achieve error detection.

Problem Statement:

An original message is given in a stream of bytes, find the value of the checksum using the CRC algorithm. Here, checksum indicates the remainder after performing CRC division on message.

After receiving the message the receiver should check whether the given message is corrupted or not by the channel. If the message is corrupted it prints an error message.

Algorithm:

• Input the message M(x) which is generated by the sender.

• Input the generating polynomial P(x) which acts as the divisor.

• If r is the order of P(x), r bits are appended to the M(x) which gives a block of bits

• Now, modulo 2 division by P(x) is performed on the block obtained earlier.

• The remainder after division is appended to M(x) which is the transmitted message T(x)

 • The receiver divides T(x) with P(x) which gives the remainder r(x).

• If the obtained r(x) is all zeros, then there is no error in the received message else an error is detected.

CODE:

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main() {
int i,j,keylen,msglen;
char input[100], key[30],temp[30],quot[100],rem[30],key1[30];
printf("Enter Data: ");
gets(input); //input message
printf("Enter Key: ");
gets(key); //input divisor
keylen=strlen(key); //length of divisor
msglen=strlen(input); //length of original message
strcpy(key1,key);
for (i=0;i<keylen-1;i++) {
input[msglen+i]='0';
```

```c
}
for (i=0;i<keylen;i++)
temp[i]=input[i];
for (i=0;i<msglen;i++) { //performing binary division
quot[i]=temp[0];
if(quot[i]=='0')
for (j=0;j<keylen;j++)
key[j]='0'; else
for (j=0;j<keylen;j++)
key[j]=key1[j];
for (j=keylen-1;j>0;j--) {
if(temp[j]==key[j])
rem[j-1]='0'; else
rem[j-1]='1';
}
rem[keylen-1]=input[i+keylen];
strcpy(temp,rem);
}
strcpy(rem,temp);
printf("\nQuotient is ");
for (i=0;i<msglen;i++)
printf("%c",quot[i]);
printf("\nRemainder is ");
for (i=0;i<keylen-1;i++)
printf("%c",rem[i]);
printf("\nFinal data is: "); //printing final message after
// appending with checksum
for (i=0;i<msglen;i++)
printf("%c",input[i]);
for (i=0;i<keylen-1;i++)
printf("%c",rem[i]);
char temp1[20];
printf("\n\nEnter received data: "); //input received data
scanf("%s",temp1);
for (i=0;i<keylen;i++)
```

```c
temp[i]=temp1[i];
for (i=0;i<msglen;i++) //division with same divisor
 {
quot[i]=temp[0]; //to check remainder is 0 or not
if(quot[i]=='0')
for (j=0;j<keylen;j++)
key[j]='0'; else
for (j=0;j<keylen;j++)
key[j]=key1[j];
for (j=keylen-1;j>0;j--) {
if(temp[j]==key[j])
rem[j-1]='0'; else
rem[j-1]='1';
}
rem[keylen-1]=temp1[i+keylen];
strcpy(temp,rem);
}
strcpy(rem,temp);
printf("\nQuotient is ");
for (i=0;i<msglen;i++)
printf("%c",quot[i]);
printf("\nRemainder is ");
for (i=0;i<keylen-1;i++)
printf("%c",rem[i]);
int flag=0; //initializing flag=0
for(i=0;i<keylen-1;i++)
 {
 if(rem[i]=='1') //if remainder contains 1 at any bit
 {
 flag=1; //flag value changes to 1
 break;
 }
 }
 if(flag==0) //the channel did not corrupt the message
 {
```

```
printf("\nNo Error");
}
else //else the message is corrupted
{
printf("\n Error");
}
}
```

OUTPUT:

TEST CASE-1:

```
Enter Data: 1001
Enter Key: 1011

Quotient is 1010
Remainder is 110
Final data is: 1001110

Enter received data: 1001110

Quotient is 1010
Remainder is 000
No Error
```

Explanation:
In the above test case, the original message given by the sender is 1001 and the divisor is 1011.CRC algorithm is computed to find remainder 110 and it is appended to the original message. Now, the transmitted message is 1001110. The receiver receives data(1001110) and should check whether it is corrupted or not so again the division is performed using the XOR operator the remainder obtained is 000 which means the message is not corrupted.

```
Quotient is 1010
Remainder is 110
Final data is: 1001110

Enter received data: 1011110

Quotient is 1000
Remainder is 110
 Error
```

Explanation:

In the above test case, the data entered by the sender is 1001 and the divisor is 1011. The remainder is computed using division and we get the remainder as 110. It is appended to the original message. Now, the final transmitted data is 1001110. The received data is 1011110. After performing the division we get the remainder as 110 which means the message was corrupted by the channel.

```
Enter Data: 1001
Enter Key: 1011

Quotient is 1010
Remainder is 110
Final data is: 1001110

Enter received data: 1001100

Quotient is 1010
Remainder is 010
 Error
```

Explanation:

The received message in these cases is 1001100. The message is not the same as the original message transmitted and the checksum obtained is also not 000. So the error is detected.

```
Enter Data: 1001
Enter Key: 1011

Quotient is 1010
Remainder is 110
Final data is: 1001110

Enter received data: 1001101

Quotient is 1010
Remainder is 011
 Error
```

Explanation:

The received message, in this case, is 1001101. The message is not the same as the original message transmitted and the checksum obtained is also not 000. So the error is detected.

TEST CASE-2:

```
Enter Data: 1011001
Enter Key: 100000111

Quotient is 1011000
Remainder is 10001000
Final data is: 101100110001000

Enter received data: 101100110001100

Quotient is 1011000
Remainder is 00000100
 Error
```

Explanation:

In the above case, the message given by the sender is 1011001 and the divisor is polynomial with bits 100000111, CRC algorithm is performed to compute the remainder and this is appended to the original message. The transmitted message now is 101100110001000. The division is performed again using the XOR operation by flipping 1 bit of the received message and the checksum is obtained as 00000100, proving there is an error in a transmitted message.

```
Enter Data: 1011001
Enter Key: 100000111

Quotient is 1011000
Remainder is 10001000
Final data is: 101100110001000

Enter received data: 111100110001000

Quotient is 1111000
Remainder is 11100000
 Error
```

Explanation:

Given, flip one bit in the message bits. If the received message now is 111100110001000, the second bit is changed. The error will be detected.

```
Enter Data: 1011001
Enter Key: 100000111

Quotient is 1011000
Remainder is 10001000
Final data is: 101100110001000

Enter received data: 101100111001000

Quotient is 1011000
Remainder is 01000000
 Error
```

Explanation:

Here, a bit is flipped in the checksum with the same message and divisor. The received message now is 101100111001000. The error is detected because the message received is not the same as the transmitted message.

```
Enter Data: 1011001
Enter Key: 100000111

Quotient is 1011000
Remainder is 10001000
Final data is: 101100110001000

Enter received data: 111100110011000

Quotient is 1111000
Remainder is 11110000
 Error
```

Explanation:

Given., flip 1 bit from the message and 1 bit from the checksum. So if the received message is 111100110011000 where the 2nd and 11th bits are changed, an error will be detected.

PROBLEMS FACED:

Starting I wasn't able to understand the CRC algorithm and wasn't able to implement the code. Computing the checksum and appending are difficult to implement for me.

CONCLUSION:

I have understood the concept of the CRC algorithm and implemented using C language. My coding skills in c language have improved. I was able to detect errors in the received message.