Introduction to Computer Networks
Machine Problem 3: Cyclic Redundancy Check
Announced: May 27 (Thu), Due: June 11 (Fri)

1. 과제의 목표

Cyclic redundancy check (CRC)를 이용하여 오류를 검출한다.

2. 작성해야 할 프로그램

crc_encoder: CRC를 이용하여 dataword를 codeword로 변환한다.
crc_decoder: codeword를 dataword로 복원하면서, 오류가 있으면 검출한다.

3. 상세설명 (주의 깊게 읽고 지시사항을 따라 구현할 것)

3.1. crc_encoder

(1) crc_encoder는 다음과 같이 실행시킨다.

```
./crc_encoder input_file output_file generator dataword_size
```

프로그램에 들어갈 인자는 4개로, 다음과 같다.
- input_file: 전송하고자 하는 파일
- output_file: 전송하고자 하는 파일을 CRC로 encode한 파일
- generator: CRC의 generator
- dataword_size: dataword의 크기. crc_encoder에서는 파일을 dataword의 크기로 나누어 각 dataword를 codeword로 변화시켜야 한다. 단위는 비트(bit)이다.

- 만약 인자의 수가 맞지 않으면 아래와 같은 메시지를 출력하고 종료한다.

```
usage: ./crc_decoder input_file output_file result_file generator dataword_size
```

(2) 만약 input_file을 open하지 못하면 다음과 같은 메시지를 출력하고 종료한다.

```
input file open error.
```

(3) 만약 output_file을 open하지 못하면 다음과 같은 메시지를 출력하고 종료한다.

```
output file open error.
```

(4) dataword_size는 4 또는 8만 지원한다. 만약 4 또는 8이 아닌 경우, 다음과 같은 메시지를 출

력하고 종료한다.

```
dataword size must be 4 or 8.
```

(5) crc_encoder는 입력 파일에서 읽어 들인 데이터를 dataword 크기로 나누어 준다.
- 예를 들어 입력 파일에 'A' 한 글자만 있다고 가정하자. 'A'는 ascii 코드로 65번이며, 2진수로 표현하면 01000001이다. 따라서 첫번째 dataword는 '0100'이며, 두번째 dataword는 '0001'이 된다.

(6) 각 dataword를 codeword로 바꿔준다. 이를 위하여 아래와 같이 modulo-2 division을 수행하여 나머지를 구한다.

- 프로그램의 인자로 주어진 generator가 '1101'이라고 가정한다.
- 첫번째 dataword는 '0100'이다.
- generator가 네 자리이므로, dataword 뒤에 '000'을 붙여준다.
- modulo-2 나눗셈을 이용하여 '0100000'을 '1101'로 나눠준다.

```
                        1  1  1
            ---------------------
1  1  0  1 | 0  1  0  0  0  0  0
               1  1  0  1
            ---------------------
                  1  0  1  0
                  1  1  0  1
            ---------------------
                     1  1  1  0
                     1  1  0  1
            ---------------------
                        0  1  1
```

- 나머지가 '011'이므로 이를 dataword 뒤에 붙여서 codeword를 만든다.
- dataword '0100'의 codeword는 '0100011'이 된다.
- 나머지 dataword도 동일한 방식으로 codeword로 바꿔준다.

(7) CRC를 이용하여 변환한 codeword들을 output file에 써 주어야 한다. 그런데 한 가지 문제는, codeword는 8의 배수가 아니기 때문에 파일의 사이즈가 바이트 단위로 떨어지지 않을 수 있다는 것이다.
- 예를 들어 'A' 한 글자로 이루어진 파일은 내용이 8비트이지만, 4비트인 generator를 써서 codeword로 바꾸면 총 14비트가 된다. 파일은 바이트 단위로 쓰기 때문에 이를 16비트 (2바이트)로 바꾸어야 하는데, 이를 위하여 0을 붙여 넣는 패딩(zero-padding)을 사용한다.
- 패딩은 앞에 넣을 수도 있고 뒤에 넣을 수도 있는데, 여기에서는 **앞에 패딩을 넣는 것으로 한다**.
- 'A'를 4비트씩 끊어서 codeword로 바꿔주면 '01000110001101'이 된다. 그럼 패딩을 두 비트 넣어줘야 16비트가 되므로 앞에 0을 두 개 붙인다. 결과적으로 '0001000110001101'이 된다. 이 중 앞의 두 비트는 codeword에 속한 비트가 아니라 패딩 비트이다.

- 수신자는 encode된 파일을 받아서 decode를 시켜야 하는데, 그러려면 패딩 비트가 몇 개인지를 알아야 한다. 이를 위하여 output file의 첫번째 바이트는 패딩 비트의 수를 나타내는 바이트가 되도록 넣는다.
- 위의 예에서 패딩 비트는 2개이기 때문에, output file의 첫번째 비트는 '00000010'이 된다.
- 결과적으로 'A' 한 글자로 이루어진 입력 파일을 crc_encoder 프로그램에 넣으면 출력 파일의 내용은 다음과 같다. 여기서는 값을 2진수로 표현하였다.

```
00000010 00010001 10001101
```

3.2. crc_decoder

(1) crc_decoder는 다음과 같이 실행시킨다.

```
./crc_decoder input_file output_file result_file generator dataword_size
```

프로그램에 들어갈 인자는 5개로, 다음과 같다.
- input_file: CRC로 encode된 파일
- output_file: CRC를 제거하고 원래 데이터 파일을 복원한 파일
- result_file: 전체 프레임의 수와 오류가 난 프레임의 수를 표시할 파일
- generator: CRC의 generator
- dataword_size: dataword의 크기. 단위는 비트(bit)이다.

- 만약 인자의 수가 맞지 않으면 아래와 같은 메시지를 출력하고 종료한다.

```
usage: ./crc_decoder input_file output_file result_file generator dataword_size
```

(2) 만약 input_file을 open하지 못하면 다음과 같은 메시지를 출력하고 종료한다.

```
input file open error.
```

(3) 만약 output_file을 open하지 못하면 다음과 같은 메시지를 출력하고 종료한다.

```
output file open error.
```

(4) 만약 result_file을 open하지 못하면 다음과 같은 메시지를 출력하고 종료한다.

```
result file open error.
```

(5) dataword_size는 4 또는 8만 지원한다. 만약 4 또는 8이 아닌 경우, 다음과 같은 메시지를 출력하고 종료한다.

```
dataword size must be 4 or 8.
```

(6) crc_decoder를 먼저 입력 파일의 첫번째 바이트를 읽어 패딩의 크기를 알아낸다.

(7) 두 번째 바이트에 있는 패딩을 제거한다.

(8) 그 이후부터 있는 비트들은 codeword의 크기로 나누어 준다.
- 예를 들어 입력 파일이 '00000010 00010001 10001101' 이면, 첫 번째 바이트에서 패딩의 크기가 2비트인 것을 알고, 두 번째 바이트의 앞의 00을 제거(무시)한 다음, 나머지를 codeword로 나눈다. 이와 같이 하면 두 개의 codeword, '0100011'과 '0001101'이 나온다.

(9) 각 codeword에 대하여, generator를 이용하여 modulo-2 나눗셈을 함으로써 codeword에 오류가 있는지 확인한다. 전체 codeword의 개수와 오류가 난 codeword의 개수를 기록한다.

(10) codeword에 오류가 있든 없든 dataword로 복원하여 출력 파일에 쓴다.

(11) result file에는 총 codeword 개수와 오류가 난 codeword의 개수를 기록한다. 예를 들어 총 codeword 개수가 23개이고 그 중에 오류가 난 codeword가 5개라면 result file에는 다음과 같이 한 줄만 쓰면 된다.

```
23 5
```

- 맨 처음에 있는 패딩 바이트와 패딩 비트들은 총 codeword의 수나 오류가 있는 codeword의 수에 포함되지 않는다.

3.3. linksim

- linksim은 숙제로 구현하는 프로그램이 아니고 바이너리 포맷으로 제공되는 프로그램이다.

- linksim은 다음과 같이 실행시킨다.

```
./linksim inputfile outputfile error_ratio(0-1) seed_num
```

- input file은 매체를 통과시키기 전의 파일이고 output file은 매체를 통과시킨 파일이다.
- error_ratio는 각 비트 별 에러율을 뜻한다. 만약 error_ratio가 0.1이라면, 어떤 비트에서 오류가 날 확률이 각각 10%라는 뜻이다. error_ratio는 0과 1 사이의 값이어야 한다.
- seed_num은 random number generator의 seed값으로 이 값을 동일하게 하면 랜덤 넘버의 시퀀스가 같고, 이 값을 다르게 하면 달라진다.

3.4. 실행 순서

crc_encoder와 crc_decoder의 구현을 마쳤으면 다음과 같이 테스트한다.
데이터가 들어있는 파일의 이름이 datastream.tx 라고 가정한다.

>> ./crc_encoder datastream.tx codedstream.tx 1101 4
>> ./linksim codedstream.tx codedstream.rx 0.0 1001
>> ./crc_decoder codedstream.rx datastream.rx result.txt 1101 4

- 여기서는 linksim의 에러율을 0으로 했으므로, 프로그램에 오류가 없다면 datastream.tx와 datastream.rx는 완전히 일치해야 한다.
- result.txt의 경우 오류가 없었으므로 두 번째 숫자는 0이 되어야 하고, 첫번째 숫자는 dataword의 크기가 4인 경우에는 최초 입력 파일 크기의 두 배, dataword의 크기가 8인 경우에는 최초 입력 파일의 크기가 된다.
- 위의 예에서 datastream.tx이 42바이트였다면, result.txt는 다음과 같아야 한다.

```
84 0
```

- 오류가 발생하도록 하려면 linksim의 error_ratio를 조정해주면 된다.

>> ./crc_encoder datastream.tx codedstream.tx 1101 4
>> ./linksim codedstream.tx codedstream.rx 0.05 1001
>> ./crc_decoder codedstream.rx datastream.rx result.txt 1101 4

이와 같이 하면 datastream.tx와 datastream.rx는 오류로 인해 달라지고, result.txt의 값은 다음과 같다.

```
84 26
```

다시 말해 총 84개의 codeword가 전송되었고, 이 중에 26개에서 오류가 검출되었다는 뜻이다.

- datastream.tx과 같은 텍스트 파일 외의 파일들도 input file로 주어질 수 있다.

4. 과제 제출

(1) 제출해야 하는 파일은 두 개이며, 그 이름은 아래와 같다. 빨간 색 부분은 자기 학번으로 변경한다. 두개의 파일을 아래의 파일명과 같이 압축해서 제출한다.

```
hw3_20190001.zip
      ├─ crc_encoder_20190001.cpp
      └─ crc_decoder_20190001.cpp
```

(2) 이 파일은 cspro에서 g++ 컴파일러로 다음과 같이 컴파일 될 예정이다.

```
g++ -Wall -o crc_encoder_20190001 crc_encoder_20190001.cpp
g++ -Wall -o crc_decoder_20190001 crc_decoder_20190001.cpp
```

- 제출하기 전에 컴파일과 실행이 잘 되는지 테스트해보고 제출한다.

(3) 아이디어는 상의할 수 있으나, 코드를 표절하지 말 것. 수강생들 사이뿐만 아니라 인터넷에 있는 코드도 동일하게 사용하지 말아야 한다. 표절인 경우에는 0점 처리된다.

(4) 지각 제출은 마감일 후 3일까지 허용하며, 하루에 10%씩 감점한다.