BTN415 Lab 9 – Bitfields and Bitwise Operators

In this lab, you will create functions that will calculate a CRC as well as a checksum of a dynamically allocated data in a user-defined struct.

LEARNING OUTCOMES

Upon successful completion of this lab, you will have demonstrated the ability to:

- Work with bitfields
- Operate on individual bytes
- Operate on individual bits

For this lab, you should create two functions, one called **calculate_packet_crc**, and another called **calculate_packet_checksum**, following the specifications below. These functions should be written at the provided spaces at the top of the source code provided in our Github repository at https://github.com/marceljar/BTN415_Labs/blob/main/lab9/main.cpp). In this source code, we define a struct packet as:

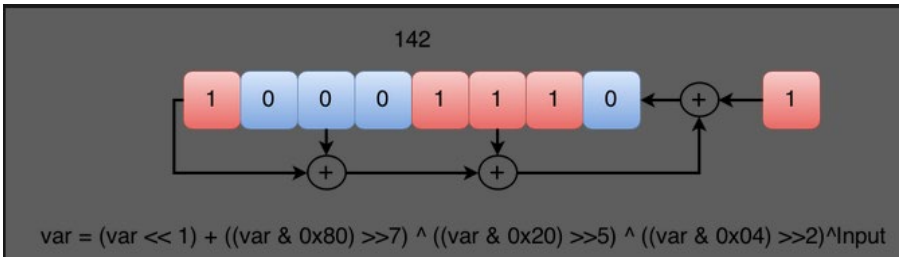
```
struct new_packet {
    char number;
    unsigned char urg : 1;
    unsigned char ack : 1;
    unsigned char psh : 1;
    unsigned char rst : 1;
    unsigned char syn : 1;
    unsigned char fin : 1;
    unsigned char : 2; //padding
    unsigned int size;
    char* data;
    unsigned char crc;
    unsigned int checksum;
};
```

A description of each function that needs to be defined, as well as the number of marks assigned to each one, is provided in what follows.

PART A – [2.5 marks]

unsigned char calculate_packet_crc(new_packet)

This function should take as an argument a new_packet. Then, it should return, as an unsigned char, the CRC of the bits in the new_packet **data** field (and only the data in the **data** field; ignore other struct fields), using a CRC circuit as shown in the image below.



PART B – [2.5 marks]

unsigned int calculate_packet_checksum(new_packet)

This function should take as an argument a **new_packet**. Then, it should calculate the checksum, as an *unsigned int,* of the data in the **new_packet data** field (and only the data in the **data** field; ignore other struct fields). Note that, to calculate the checksum, you should treat the content in your **data** field as integers. I.e., you should add the bytes in it, grouping them four by four and read these 4 bytes (32 bits) as an integer. *Hint: Use an auxiliar int pointer that starts by pointing at my_packet.data. Beware that if the data has a number of bytes that it is not a multiple of 4, you will have to create a special mechanism to deal with the last bytes (which could be 1, 2, or 3).*

SUBMISSION INSTRUCTIONS

You only need to submit the main.cpp file.