Successfully implemented calculator:

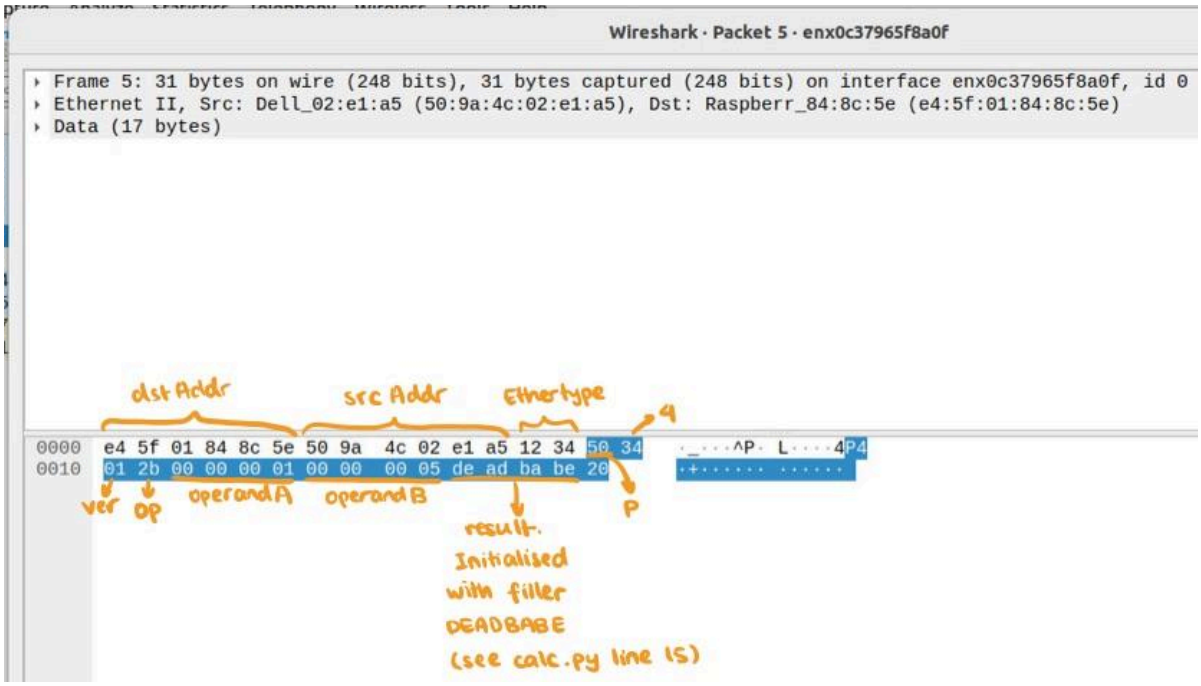
```
ubuntu@ubuntu:~/Desktop/CWM-ProgNets/assignment5$ sudo python3 calc.py
> 1 + 5
1 + 5
6
> 1 - 5
1 - 5
4294967292
> 1 + 5
1 + 5
6
> 9 + 5
9 + 5
14
> 0 + 1
0 + 1
1
> 1 - 0
1 - 0
1
> 5 - 2
5 - 2
3
> 1 & 0
1 & 0
0
> 2 & 0
2 & 0
0
> 2 & 1
2 & 1
0
> 0 | 1
0 | 1
1
> 1 | 1
1 | 1
1
> 1 ^ 1
1 ^ 1
0
>

Expected number literal.
> 1 ^ 0
1 ^ 0
1
```

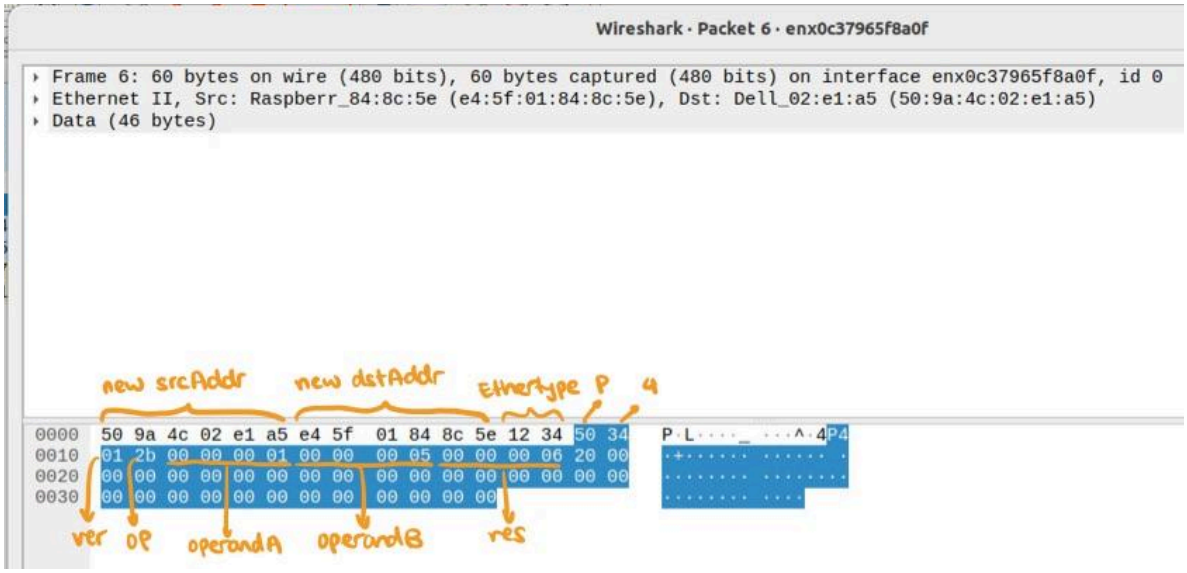
```
> 1 + 5
1 + 5
6
> 5 - 1
5 - 1
4
> 1 ^ 0
1 ^ 0
1
> 1 ^ 1
1 ^ 1
0
> 1 | 1
1 | 1
1
> 1 & 0
1 & 0
0
> 1 & 1
1 & 1
1
> quit
ubuntu@ubuntu:~/Desktop/CWM-ProgNets/assignment5$ cd ..
```

```
> 1 % 5
1 % 5
Expected binary operator '-', '+', '&', '|', or '^'.
> 0 - 0
0 - 0
0
> 0 + 0
0 + 0
```

Tested some edge cases, such as 0 - 0, 0 + 0, and the correct error messages are displayed when the input isn't a number (expected number literal) and the operator isn't one of the 5 supported operators (expected binary operator).

I also investigated the packets through wireshark (example of 1 + 5 = 6):





We can see the structure of the packets sent, and how the sequence of headers we define in calc.p4 is reflected in the hexadecimal representation of the packet. It is interesting to see that the return packet is padded with extra zeros (29 extra bytes of zeros)