Exercise 6

The aim of the project was to be able to play blackjack on the network, having the entire game on the Raspberry Pi. The Raspberry Pi would act as the dealer and give the player a choice of ‘hit’ or ‘stand’, which would then be sent in a packet, and the Pi replies by showing cards if ‘hit’ was selected or showing the dealer's hand if ‘stand’ was chosen. It would then include the result of the game, and if the player had gone bust or the dealer had. The player started with the first 2 cards, the dealer with the 15th, and their respective counts reflect this. Due to the lack of a random number generator in $P4_{16}$, this aspect had to be done on the Python side. The Python code also returns the cards to the player and contains the win messages upon receiving the bust and result bits from the return packet.

Architecture and operation
The P4 code contains two headers: Ethernet (112 bits) and Blackjack (132 bits).
The metadata is all standard metadata.
For the parser, I used the same as the example calc.p4 to extract the Ethernet header and check and extract the Blackjack header. I used the checksum verification from calc.p4 also.

For the ingress processing, I had several actions:

Send_back switches the MAC addresses and sends the packet back through the same port.

Check_sum_dealer and Check_sum_player both sum the card values and check if either is bust. Additionally, check_sum_dealer will stop the dealer if the sum is over 17, as is traditional. There have been some issues with assigning values, which caused the player to lose immediately upon ‘hitting’.

There are two commented actions, new_card and check_card, which would’ve been used to generate cards on the p4 side had a random number been supported.

The two main actions are operation_hit and operation_stand.
Operation_hit increases the player count and calls check_sum_player, which shows the player another card and checks if they are bust.
Operation_stand calls new_dealer_card, which simply calls check_sum_dealer and changes the dealer count. It repeats this until the dealer reaches 17 or 21, and the bust value changes, causing send_back to be called.

Operation_drop is standard and drops the packet.

Operation_hit and operation_stand are both listed in the table as actions with the move as the key. Operation_drop is the default option, so packets without proper inputs are not returned.

Egress processing, Deparser, and Switch are all the same as calc.p4.

Testing:

In some initial tests, it was able to produce the first 2 cards when a packet was sent, but any other input would cause the loss message. When attempting to fix this, there was an error in the Python code that was not fixed in time. This is why there is a second file called blackjack_org.py that holds a previous version. Unfortunately, under time pressure, no screenshots were captured of it running or any errors.