Problem Solving 1

When playing a game of solitaire, as with most card games, it is necessary to deal out some number of cards to start a game. Different games require different starting "hands".

In the space to the right, write a function called "deal_hand" that makes a standard deck of cards, deals out 4 cards for the starting hand, and returns the hand.

```
import random

def deal_hand():

        (deck) = ([f"{rank} of {suit}" for rank in range(1, 14) for suit in ["Hearts",
"Diamonds", "Spades", "Clubs"]])
        (hand) = deck([:4])
        return (hand)

def main():

        (hand) = deal_hand()
        print(hand)

main()
```

```
import deal_hand()

def test_deal_hand():

        (hand) = deal_hand()
        assert len(hand) == (4)
        assert all((card, str) for card in hand)

def main():

        test_deal_hand()

main()
```

Problem Solving 2

When doing iterative development, it is important to create unit tests to confirm your code is not only working, but continues to work as your development progresses.

The function that you wrote in the previous problem created a standard deck of cards and dealt out some number of cards. What tests can you do at this point?

In the space to the left, write at least one test to confirm that your "deal_hand" function is working properly.

Problem Solving 3

Discarding is when you remove cards from your hand and place them in a “waste” or “discard” pile. Specific games vary in what card(s) get discarded when. You may even have to discard cards that are in a specific location.

In the space provided, write a function called “discard”, that takes a hand and a number which represents how many cards to discard. You will discard the last n cards in the player’s hand using slicing. Return the hand.

```
def discard(hand, cards):

        if (cards) > (len(hand)):
                raise ValueError("Invalid Input")
        else:
                (discarded) = (hand[-num_cards:])
                pop.(hand[-num_cards:])
                return (hand)

def main():

        test = discard([“11 of Clubs”, “3 of Hearts”, “8 of Hearts”, “12 of Clubs”], 2)
        print(test)

main()
```

```
def discard(hand, cards, location):

        if (cards) > (location):
                raise ValueError("Invalid Input)
        if (location) > (len(hand)):
                raise ValueError("Invalid Input)
        else:
                (discarded) = (hand[location-cards:location])
                pop.(hand[location-cards:location])
                return (hand)

def main():

        test = discard([“11 of Clubs”, “3 of Hearts”, “8 of Hearts”, “12 of Clubs”], 2, 2)
        print(test)

main()
```

Problem Solving 4

In the space to the left modify your “discard” function to pass a third parameter called location that specifies how many cards from the end (the right side of the hand) to keep before discarding the number of cards from the hand. As before, return the hand.

Problem Solving 5

What could you use to validate that your updated discard function is working properly?

Write at least one unit test to verify that your updated discard function works properly.

```
import discard()

def test_discard():

        (hand) =([“11 of Clubs”, “3 of Hearts”, “8 of Hearts”, “12 of Clubs”])
        (expected_hand) = ([“11 of Clubs”, “3 of Hearts”])
        assert discard(hand, 2, 3) == (expected_hand)

        (hand) = ([“11 of Clubs”, “3 of Hearts”, “8 of Hearts”, “12 of Clubs”])
        try:
                discard(hand, 3, 2)
                assert (False)
        except (ValueError):
                assert (True)

def main():

        test_discard()

main()
```