ECS 140A Homework 6 – Problem 2

1 Prolog

Step 3: Working Code

```
is_palindrome([]).
is_palindrome(X) :- reverse(X, Y), reverse(Y, X).
```

Step 4: Debug Process

Bug 1

```
?- is_palindrome([a, b]).
true.
```

`reverse(X, Y)` returns true if Y is the reverse of X, so my code would return true in any case. I need to also check that the items in the reverse are in the same order as the original.

Fixed code:

```
is_palindrome(X) :- reverse(X, Y), is_eq(X, Y).

is_eq(X, X).
is_eq([H1, X], [H2, Y]) :- is_eq(H1, H2), is_eq(X, Y).
```

Bug 2

```
?- is_palindrome([a, b, c, b, a]).
true ;
false.
```

Palindromes will result in true or false, instead of only true. I misused the list slicing syntax, originally wanting to compare character by character of the list and reverse list. However, I realized I overthought the logic since a palindrome is just the same backwards as it is forwards so a one-liner using `reverse()` will suffice.

Fixed code:

```
is_palindrome(X) :- reverse(X, X).
```

Results of given tests:

```
?- is_palindrome([a, b, c, b, a]).
true.

?- is_palindrome([]).
true.
```

Results of additional tests:

```
?- is_palindrome([a]).
true.

?- is_palindrome([a, a]).
true.

?- is_palindrome([a, b]).
false.

?- is_palindrome([a, b, a]).
true.

?- is_palindrome([r, a, c, e, c, a, r]).
true.

?- is_palindrome([r, a, c, e]).
false.

?- is_palindrome([1, 2, 3]).
false.

?- is_palindrome([1, 2, 3, 2, 1]).
true.
```

Step 5: Add Documentation

```
is_palindrome(X) :- reverse(X, X). % check if X is the reverse of itself
```

Step 6: Extra Test Cases Used

- `is_palindrome([a])`: true
- `is_palindrome([a, a])`: true
- `is_palindrome([a, b])`: false
- `is_palindrome([a, b, a])`: true
- `is_palindrome([r, a, c, e, c, a, r])`: true
- `is_palindrome([r, a, c, e])`: false
- `is_palindrome([1, 2, 3])`: false
- `is_palindrome([1, 2, 3, 2, 1])`: true