Homework 9

1.

I think that as soon as the result of the expression is determined, the rest of the expression will not be evaluated. This is demonstrated with the pseudocode below.

```
if e():
    do_something()
else:
    if f() and g():
        do_something()
    else:
        if h():
            do_something()

if e() and f():
    do_something()
else:
    if g() and h():
        do_something()

if e():
    if f() or g():
        do_something()
```

2.

a)

```
def __iter__(self):
    for bucket in self.array:
        while bucket is not None:
            yield bucket.value
            bucket = bucket.next
```

b)

```
class HashTableIterator:
    def __init__(self, hash_table):
        self._hash_table = hash_table
        self._index = 0
        self._node = self._hash_table.array[self._index]

    def __next__(self):
        while self._node is None:
            self._index += 1
            if self._index >= len(self._hash_table.array):
                raise StopIteration
            self._node = self._hash_table.array[self._index]

        value = self._node.value
        self._node = self._node.next

        return value
```

c)

```
tab = HashTable(10)
for value in tab:
    print(value)
```

d)

```
tab = HashTable(10)
itr = iter(tab)
while True:
    try:
        value = next(itr)
        print(value)
    except StopIteration:
        break
```

e)

```
def forEach(self, function):
    for value in self:
        function(value)
```

3.

a)

X = green

b)

false

c)

Q = tomato, and then Q = beet

d)

return every possible combo

4.

a)

```
likes_red(X) :- likes(X, Y), color(Y, red), food(Y).
```

b)

```
likes_foods_of_colors_that_menachen_likes(X) :- likes(menachen, Y), color(Y, Color), food(Y), likes(X, Z), color(Z, Color), food(Z).
```

5.

```
direct(X, Y) :- road_between(X, Y).
direct(X, Y) :- road_between(Y, X).

reachable(X, Y) :- direct(X, Y).
reachable(X, Y) :- direct(X, Z), reachable(Z, Y).
```

6**.**

1,3,4,6,7,8 all unify

2 doesnt because the number of args for the both are different

5 doesn’t because the second args are different

8 doesn’t because z cannot be both bletch and barf

10 doesn’t a cannot be both an atom and list

7.

X

Y

T, NT, X

8.

Acc

Tail, Total

1

Sum1

9.

```
gen_list(_, 0, []).


gen_list(Value, N, [Value|Tail]) :-
    N > 0,
    N1 is N - 1,
    gen_list(Value, N1, Tail).
```

10.

```
append_item([], Item, [Item]).

append_item([Head|Tail], Item, [Head|ResultTail]) :-
    append_item(Tail, Item, ResultTail).
```