Homework 9

Tejas Kamtam

305749402

CS 131 - Fall 2023

Problem 1

An example like this will execute the actual functions in order, left-associative so we don't have to worry about side effects. Within the conditional expressions, the `AND` operator takes precedence so the first conditional will evaluate `e() or f()` first and then `g() or h()`. Within these, the `OR` will short-circuit if the left value is True, but the `AND` will execute both sides.
The second expression similarly, will short-circuit if `e() and f()` is True. The expression itself will short-circuit if `e()` is False.
Finally the last expression will short cirucit if `f() and g()` evaluates to False, since the paretheses takes precedence.

Problem 2

Problem a

An external generator function:

```
def iterate(ht):
  for i in range(len(ht.array)):
    node = ht.array[i]
    while node:
      yield node.value
      node = node.next
```

Problem b

The new class with the `__iter__` method:

```
class HashTable:
  def __init__(self, buckets):
    self.array = [None] * buckets
  def insert(self, val):
    bucket = hash(val) % len(self.array)
    tmp_head = Node(val)
    tmp_head.next = self.array[bucket]
    self.array[bucket] = tmp_head
  def __iter__(self):
    for i in range(len(self.array)):
      node = self.array[i]
      while node:
        yield node.value
        node = node.next
```

Problem c

Using the generator:

```
# initialize the hash table
h = HashTable(3)
for i in range(10):
  h.insert(i)

for i in iterate(h):
  print(i)
```

Using the class:

```
# initialize the hash table
h = HashTable(3)
for i in range(10):
  h.insert(i)

for i in h:
  print(i)
```

Problem d

```
i = iter(h)
while True:
  try:
    print(next(i))
  except StopIteration:
    break
```

Problem e

```
def forEach(self, f):
  for i in range(len(self.array)):
    node = self.array[i]
    while node:
      f(node.value)
      node = node.next
```

Problem 3

Problem a

```
X = green
```

Part b

```
false
```

Part c

```
Q = tomato
Q = beet
```

Part d

```
Q = celery, R = green
Q = tomato, R = red
Q = persimmon, R = orange
Q = beet, R = red
Q = lettuce, R = green
```

Problem 4

Part a

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

Part b

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

Problem 5

```
reachable(A,B) :- road_between(A,B).
reachable(A,B) :- road_between(B,A).
reachable(A,B) :- road_between(A,C),road_between(C,B).
```

Problem 6

1. true, `{X→ bar}`
2. fase, the 2 facts have different arity
3. true, `{X⟷Z}`
4. true, `{X→barf, Y→bletch}`
5. false, the second atoms in the fact are not the same
6. true, `{X→bar, Y→barf}`
7. true, `{Y→bar(a,Z)}`
8. false, the matchings for `barf` are not one-to-one
9. true, `{Q→[A,B|C]}`
10. false, `X` matches to multiple different atoms

Problem 7

```
% adds a new value X to an empty list
insert_lex(X,[],[X]).
% the new value is < all values in list
insert_lex(X,[Y|T],[X,Y|T]) :- X =< Y.
% adds somewhere in middle
insert_lex(X,[Y|T],[Y|NT]) :- X > Y, insert_lex(X,T,NT).
```

Problem 8

```
% count_elem(List, Accumulator, Total)
% Accumulator must always start at zero
count_elem([], Total, Total).
count_elem([_|Tail], Sum, Total) :-
Sum1 is Sum + 1,
count_elem(Tail, Sum1, Total).
```

Problem 9

```
gen_list(_,0,[]).
gen_list(W,N,[W|T]) :- N > 0, NL is N - 1, gen_list(W,NL,T).
```

Problem 10

```
append_item([],X,[X]).
append_item([H|T],X,[H|NT]) :- append_item(T,X,NT).
```