

## Question 1

The following if blocks are equivalent to the one given in the problem statement:

**I.**

```
if (e())
    do_something();
if (f())
    if (g())
        do_something();
if (h())
    do_something();
```

**II.**

```
if (e())
    if (f())
        do_something();
if (g())
    if(h())
        do_something();
```

**III.**

```
if (e())
    if (f())
        do_something();
    if (g())
        do_something();
```

## Question 2

```
(a) class Node:
    def __init__(self, val):
        self.value = val
        self.next = None

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 gen(arr):
    for i in range(arr):
        curr = arr[i]
        while arr[i] is not None:
            yield curr.value
            curr = curr.next

(b) def HTIterator:
    def __init__(self, arr):
        self.arr = arr
        self.bucket = -1
        self.curr = None

    def __next__(self):
        while self.curr is None:
            self.bucket += 1

            if len(self.arr) < self.bucket:
                raise StopIteration

            self.curr = self.arr[self.bucket]
            value = self.curr.value
            self.curr = self.curr.next
            return value

(c) hash_table = HashTable(10)
    ...
    for item in hash_table:
        print(item.value)

(d) hash_table = HashTable(10)
    ...
    it = hash_table.__iter__()
    while True:
        try:
            print(it.__next__())
        except StopIteration:
            pass

(e) class HashTable:
    ...
```

```
def forEach(f):  
    for i in range(0, len(self.array)):  
        curr = arr[i]  
        while curr is not None:  
            f(curr)  
            curr = curr.next
```

### Question 3

- (a) `X = green`
- (b) `false`
- (c) `Q = tomato,`  
`Q = beet`
- (d) `Q = celery, R = green,`  
`Q = tomato, R = red,`  
`Q = persimmon, R = orange,`  
`Q = beet, R = red,`  
`Q = lettuce, R = green`

## Question 4

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

(b) `likes_food_color(X, Z) :-  
 food(Y),  
 likes(X, Y),  
 color(Y, Z).  
likes_foods_of_colors_menachen_likes(X) :-  
 likes_food_color(X, Z),  
 likes_food_color(menachen, Z).`

## Question 5

```
reachable(X, Y) :-  
    road_between(X, Y);  
    road_between(Y, X);  
    road_between(X, Z),  
    road_between(Z, Y).
```

## Question 6

- (a)  $\{X \rightarrow \text{bar}\}$
- (b) Does not unify since the arities do not match.
- (c)  $\{Z \leftrightarrow X\}$
- (d)  $\{X \rightarrow \text{barf}, Y \rightarrow \text{bletch}\}$
- (e) Does not unify since  $\text{bletch} \neq \text{barf}$
- (f)  $\{X \rightarrow \text{bar}, Y \rightarrow \text{barf}\}$
- (g)  $\{Y \rightarrow \text{bar}(a, Z)\}$
- (h) Does not unify since  $Z$  cannot bind to both  $\text{barf}$  and  $\text{bletch}$ .
- (i)  $\{Q \rightarrow [A|B|C]\}$
- (j) Does not unify since  $X$  cannot bind to  $[a]$ .