The Python program below implements a solver for the reaching-definition equations on the right. We say that the function F has a fixed point X if X = F(X). Let's see if "solve" below has a fixed-point.

```
def eval(IN1, OUT1, IN2, OUT2, IN3, OUT3, IN4, OUT4):
   _IN1 = set()
   _OUT1 = IN1.difference([1, 3, 4]).union([1])
   _IN2 = OUT1.union(OUT2)
   _OUT2 = IN2
   _IN3 = OUT2
   _OUT3 = IN3.difference([1, 3, 4]).union([3])
   _IN4 = OUT2
   _OUT4 = IN4.difference([1, 3, 4]).union([4])
   return (_IN1, _OUT1, _IN2, _OUT2, _IN3, _OUT3, _IN4, _OUT4)

def solve(sets):
   new_sets = eval(*sets)
   while (new_sets != sets):
      new_sets = sets
      sets = eval(*new_sets)
   return sets

print(solve([set() for i in range(8)]))

print(solve([set([1, 5]) for i in range(8)]))
```

**Figure 1**: Implementation of an iterative solver for the data flow equations in Figure-2.

$IN[1] = \{\}$

$OUT[1] = (IN[1] \setminus \{1, 3, 4\}) \cup \{1\}$

$IN[2] = OUT[1] \cup OUT[3]$

$OUT[2] = IN[2]$

$IN[3] = OUT[2]$

$OUT[3] = (IN[3] \setminus \{1, 3, 4\}) \cup \{3\}$

$IN[4] = OUT[2]$

$OUT[4] = (IN[4] \setminus \{1, 3, 4\}) \cup \{4\}$

**Figure 2**: Example of data-flow equations that solve an instance of reaching-definition analysis. We omit the program that gave origin to these equations.

1. What would be printed by the first call to solve in Fig-1?

2. What would be printed by the second call to solve in Fig-1?

3. Does the "solve" function always reach a fixed point, regardless of its input?