Samin Islam
UTEPID - 80817875

1. Fill in the table below with the necessary information for each type of analysis.

	Domain	Direction	Init	Merge	Transfer
Reaching definitions	Sets of definitions	Forward	OUT[ENTRY] = ∅ OUT[*] = ∅; where * is all those basic blocks other than ENTRY	Union	$f(in_b) = use_b \cup (in_b - kill_b)$
Live variables	Sets of variables	Backward	IN[EXIT] = ∅ IN[*] = ∅; where * is all those basic blocks other than EXIT	Union	$f(out_b) = use_b \cup (out_b - kill_b)$
Constant propagation	Valuation or T	Forward	OUT[ENTRY]={v↦⊥ | v∈vars} which means: all program variables are initialized to ⊥ (undefined) at the entry.	Intersection	$f(out_b) = (in_b - kill_b) \cup gen_b$
Available expressions	Sets of expressions	Forward	OUT[ENTRY] = ∅ OUT[*] = ∅; where * is all those basic blocks other than ENTRY	Intersection	$f(in_b) = use_b \cup (in_b - kill_b)$

2. Write a convincing argument that the worklist algorithm is guaranteed to converge to a solution, given a certain condition. Be sure to state that condition. You do not need to use lattice theory -- you may if you wish but you can also just give a convincing logical argument.

- The worklist algorithm is guaranteed to converge as long as the data-flow values change in only one direction according to a clear order— this property is called monotonicity. If the facts are sets, we can order them by the subset relation (⊆). In a may analysis (like Reaching Definitions) each step can only add elements, so the sets grow. In a must analysis (like Available Expressions) we start from the universal set and each step can only remove elements, but still never reverse direction.

  Because the set of possible facts is finite and each block's value can only move one way, every block can change only a finite number of times. The worklist eventually runs out of changes, and at that point we have reached a fixed point, so the algorithm terminates and converges. For Live-Variable analysis (a backward problem) we begin with the empty set at every block. Each iteration adds variables that are used along some path. There are only finitely many variables, so the sets can grow only so far, and the process stops.