**Problem 1**

**Part a**

Bit vector framework is the Gen and Kill functions, while the live variable analysis also based on Gen and Kill

Live variable analysis is bit vector framework because it use Gen and kill.

**Part b**

Bit Vector Frameworks are distributive i.e if the flow functions f:L->L of a framework can be expressed as

f(x)=(x-Kill) U Gen where Kill, Gen ∈ L then ∀x,y ∈ L : f(x∩y)=f(x) ∩ f(y)

This follows that bit vector framework is also monotonic.

Since all Gen/Kill problem are distributive, it means all bit vector frameworks are distributive.

**Part c**

Constant propagation is an example of non distributive framework but it is bit vector framework

In general, analysis of what the program computes is not distributive.

MOP for In(s) will always be ⊑ iterative dataflow solution.

**Problem 2**

**Part a**

**partially available expressions:-**

**For reading clarity, point “l” is changed to “P”.**

An expression is partially available at a program point P, if the expression is available along some path to P.

Genn = { e | expression e is evaluated in basic block n and this evaluation is not followed by a definition of any operand of e}

Killn = { e | basic block n contains a definition of an operand of e}

**Part b**

An expression e is anticipable at a program point p, if every path from p to the program exit contains an evaluation of e which is not preceded by a redefinition of any operand of e.

**Part c**

**To remove PRE**

1. Identify partial redundancies

2. Identify program points where computations can be inserted

3. Insert expressions

4. Partial redundancies become total redundancies =⇒ Delete them.

**Placement Possible Analysis**

An expression can be safely inserted at a program point p if it is

* If it is available at p, then there is no need to insert it at p.
* If it is anticipable at p then all such occurrence should be inserted to p.

An expression should be inserted to p provided it can be inserted to p along all paths from p to exit.

* safety of Inserting to the exit of a block.

Should be inserted only if it can be inserted to the entry of all successors.

* safety of Inserting to the entry of a block.

Should be inserted only if it is upwards exposed,

it can be inserted to its exit and is transparent in the block .

* Desirability of inserting to the entry of a block.

Should be inserted only if

* it is partially available,
* For each predecessor

1. it is inserted to its exit,
2. is available at its exit.

Partial availability

Total availability

**PRE Data Flow Equation**

**Part d**

**insert l:-**

An expression is inserted at the exit of node n if

* it can be placed at the exit of n, AND
* it is not available at the exit of n, AND
* it cannot be Inserted out of n, OR it is modified in n.

Insert n = Out n ∩ (¬AvOut n) ∩ (¬Inn ∪ Kill n)

**delete l:-**

An expression is redundant in node n if

* it can be placed at the entry of n, AND
* it is upwards exposed in node n.

Redundant n = Inn ∩ AntGenn