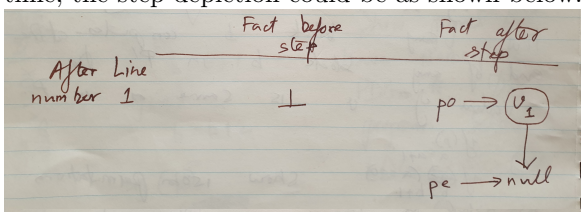
Program Analyis and Verification
Assignment 4, *Pointer analysis and slicing*
Due date: Nov. 7th, 11.59 pm.

Problem 1. This problem is about points-to analysis. Consider the program below:

```
i = 0; pe = null; po = null;
while (i < 3) {
 if (i % 2 == 0) {
  po = malloc();
  *po = pe;      // 1
 }
 else {
  pe = malloc(); // 2
  *pe = po;
 }
 i++;
}
```

1. Using the AbsDataState lattice introduced in class, and using the most-precise transfer functions, illustrate the steps in a run of Kildall's algorithm. Show in your answer only those steps that produce joined values at the points right *below* the line labeled "1" and the line labeled "2". For each step, show the existing data-flow fact at the point just before the step and the updated (i.e., joined) value at the same point after the step. Show each abstract data-flow fact as a graph (known as a *points-to graph* in the literature). For instance, for the step that visits Line 1 for the first time, the step depiction could be as shown below.



   Use V_1 for the first malloc statement in the code, and V_2 for the second. Note that in general in a points-to graph a variable or node could point to multiple nodes. Ignore the variable i in the abstract facts.

2. Show the final points-to graph at the end of the program after Kildall's algorithm terminates.

3. Assume that 101 and 103 are the concrete addresses that can be allocated at the first malloc site and 102 is the concrete address that can be allocated at the second malloc site. Show the gamma image of the Kildall solution at the end of the program. Depict each concrete state in the gamma image as a "concrete" graph (in this graph each variable or node can point to exactly one node or to null). Indicate with a tick mark the concrete states that belong to the collecting semantics and with an "X" mark the concrete states that do not belong to the collecting semantics.

4. Assuming there is no memory-allocation, show by example that the most-precise transfer function of the statement "*v1 := v2" is non-distributive. Hence, show a small program where Kildall's algorithm computes a strict over-approximation of the JOP.

Page limit for this problem is 3 pages.

Problem 2. Consider the program below. Show the PDG of the program, assuming the final-use nodes are "FinalUse(sum)" and "FinalUse(prod)". Make sure to include all appropriate annotations in each PDG edge to fully indicate its type, category, sub-category, etc. Mark with a tick the PDG nodes that belong to the slice as per the PDG-based slicing method, treating the node "FinalUse(sum)" as the criterion. Page limit: 1/2 page.

```
sum = 0;
while(i < 100) {
 if (..)
  i = i + j;
 if (..)
  j = j + i;
 prod = prod * i;
 sum = sum + i;
 i = i + 1;
}
```

Problem 3. Consider a family of programs, where each program is a sequence of *blocks*. Each block is an "if" statement, with a single assignment statement in the "then" part, and an empty "else" part. The sample program below belongs to this family, and consists of four blocks:

```
if (p)
  a = a + 1;
if (q)
  b = b + 1;
if (a < 10)
  b = b + 2;
if (a < 20)
  c = c + 1;
```

Let P be a given program in the family, and let P' be any other program in the family such that the PDG of P' is isomorphic to the PDG of P.

1. Give an argument for why P' must be a permutation of the blocks in P.

2. Show all possible programs whose PDGs are isomorphic to the PDG of the program shown above. You may show the permutations compactly by referring to the first block in the code above as b_1, the second block as b_2, and so on.

3. If a block b_j in P uses a variable v (in its condition or in the RHS of its assignment statement), prove that the blocks that define v and that come before b_j in P are exactly the blocks in P' that come before b_j and that define v. Moreover, prove that these defining blocks appear in the same order in both P and P'.

4. Using the result above, prove using mathematical induction that for any block b_j, the "if" condition in the block evaluates to the same value (true/false) and the assignment statement within the block assigns the same value to the LHS in both P and P' assuming both these programs are executed using the same initial state. You must state what the induction is on, what the inductive hypothesis is, and argue the base case and the inductive case.

Page limit is 3 pages.