<u>**SUBMITTED BY : CS22MTECH02002 (Shikhar Jain)**</u>

1) Can Index Set Splitting (ISS) pass optimize the performance of the loop? When it is helpful, give some examples.

   Ans) Yes ISS can enhance the performance of the code by basically splitting the loop w.r.t index set.
   A concrete example could be :

```
for(i=0; i<100; i++) {
if(i < m)
A[i] = A[i] * 2;
else
A[i] = A[i] * 5;
B[i] = A[i]*A[i];
}
```

   This is taken from "Generalized Index-Set Splitting" by Christopher Barton1, Arie Tal et al.
   Clearly if we are able to transform/divide this one loop into two loops such that branching is eradicated than performance will be enhanced {on the cost of may be somewhat increased code size and added loop jumps}

   The above given example is correctly transformed as below:

```
for(i=0; i<min(m,100); i++) {
A[i] = A[i] * 2;
B[i] = A[i]*A[i];
}
for(i=max(m,0); i<100; i++) {
A[i] = A[i] * 5;
B[i] = A[i]*A[i];
}
```

   It might be noticable that this transformation is slightly differnt from what has been expected in the assignment
   Cause generally this tranforamtion requires a "splitting point" which is choosen to optimize certain objective.

   Parallel execution of cloned loops can happen if these clones are generated "mindfully".

   Even different types/shapes of cache blocking[Rectangular/trapezoidal] could be applied on different clones accordingly.

SUBMITTED BY : CS22MTECH02002 (Shikhar Jain)

Ex: Before transformation:

```
for (i = 0; i < N; i++) {
       for (j = 0; j < M; j++) {
         A[i][j] = B[i][j] + C[i][j];
        }
      }
```

After transformation:

```
//------- decide stride -------
for (ii = 0; ii < N; ii += b) {
  for (jj = 0; jj < M; jj += s) {
  //-----------------------------

  //------------ iterate over the current window ----------
    for (i = ii; i < min(ii + b, N); i++) {
      for (j = jj; j < min(jj + s, M); j++)
 // ---------------------------------------------
      {
            A[i][j] = B[i][j] + C[i][j];
      }
    }
  }
            }
```

Here basically one big recatngular iteration space is divided in many smaller rectangular iteration spaces . This increases data locality as now less cache misses will happen.

Q2) Can ISS pass result in performance degradation? Give some examples.

Ans) One obvious degradation is in code size . Since at the end of this transformation we are cloning the loop body multiple times hence we are increasing no. of loop headers, exit blocks, latch blocks etc.

Ex: for(i=0;i<N;i++) { some task } ----> (for(i=0;i<N/m;i++) { some task }) m times . Here until the loop body gets changed due to change in parameters [after ISS] no performance enhancement will be observed.
Also it might happen due to such transformation,that inorder to maintain semantics, yu have to maintain buffers after each loop so that other clones could use them thus extra memory will be consumed

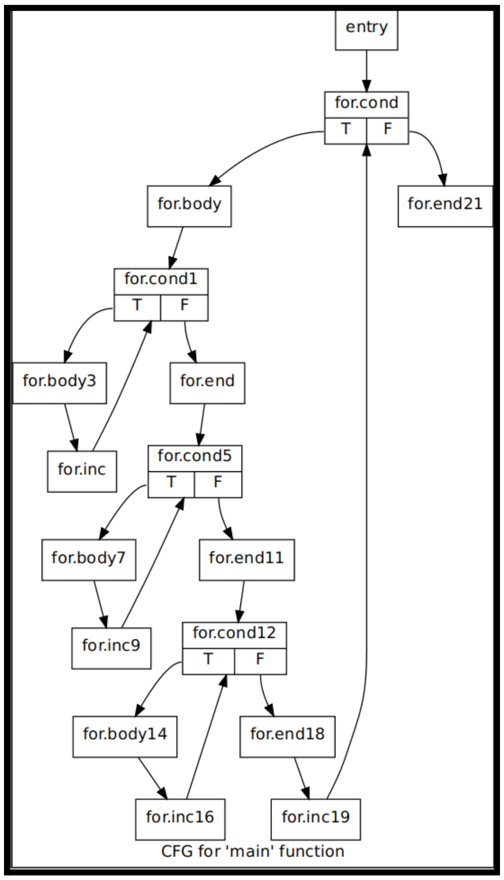
SUBMITTED BY : CS22MTECH02002 (Shikhar Jain)

SInce in naive ISS implementation parallel copies of same loop is introduced hence if these bodies are not executed parallely then will cause performance degradation due to more jumps/branchings and associated overhead.

Q3) Why do we need Loop simplify pass before ISS pass? (Do not write what Loop Simplify does. It is given in above steps :)

ISS pass requires/assumes clear “natutal loop” structure of the loop nest before proceeding . the reason for this a) to get loop induction variable and associated loop bounds b) to get loop’s exit,latch,pre-header blocks since parallel copies are introduced accordingly.

As can be seen by CFG on right how parallel Loops are connected to eachother.
Each parallel loop’s exit block dominates the pre-header/header of next immediate parallel Copy. this recursive structure goes until the last Parallel loop’s exit block follows parent loop’s Latch lock.



CFG for 'main' function

Q5) Assume Loop Simplify pass is not available. Now, how can you implement ISS pass? Write an algorithm.

If loop simplify is not available then for each loop we need to insert loop preheader and exit block at least and then properly adjust/redefine the dominance relationship accordingly We may need to do so each time we are changing the CFG due to insertion of cloned loops.

<u>SUBMITTED BY : CS22MTECH02002 (Shikhar Jain)</u>

Q6) In general, can ISS expose more opportunities for optimizations? Which optimizations would benefit by performing ISS?
Yes it can. Loop unrolling, loop parallalization/vectorization, loop tilling it can positively affect speculative execution. Small index set can correlate positively (w.r.t performance) with underlying architecture.