1. [15 points] Data dependence analysis and unimodular transformations

   ```
   for (i=3; i<N; i++) {
     for (j=0; j<M; j++) {
       A[i] = B[i-3][j+1] - 99;
       B[i+2][j+1] = sin(i * j * 3);
     }
   }
   ```

(a) For the above program, what is the direction vector for the output dependences between writes to A[i]? (Hint: Recall that (\ast, \lt), (\ast, \eq), and (\ast, \gt) are not legal dependence vectors.) Show the setup for the data dependence problem and how you derive the direction vector. (Hint: you might want to use an eqnarray in latex to list the constraints.)

(b) For the above program, what is the distance vector for the flow dependence? Show the setup for the data dependence problem and how you derive the distance vector.

(c) Can we parallelize the i loop or the j loop. Please explain why or why not for each loop.

(d) Is the problem 1 loop fully permutable? Why or why not? What is the unimodular transformation matrix that specifies a permutation of the i and j loops in the program for problem 1? Describe the legality or illegality of permutation by applying the unimodular transformation matrix to the direction vectors.

2. [10 points] Loop Fission and the Kelly and Pugh Transformation Framework

(a) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework notation. To do this, set up the dependence analysis problem, find the lexicographically non-negative dependence relation, show the fission representation using a mapping, and then show how you check the legality.

   ```
   for (i=0; i<N; i++) {
     A[i] = ...;
     ... = A[i + 1];
   }
   ```

(b) Show whether loop fission is legal or illegal for the following program using the K&P transformation framework. To do this, set up the dependence analysis problem, find the lexicographically non-negative dependence relation, show the fission representation using a mapping, and then show how you check the legality.
for (i=2; i<N; i++) {
    A[ i ] = ... ;
    ... = A[ i - 2 ];
}

(c) For both part (a) and part (b) indicate whether the data dependence relation represents a flow, anti, or output dependence.

3. [30 points] Data dependence analysis
For each of the below loops show the memory data dependence relation with all of the inequalities and equalities for all of the flow, anti, and output dependences. This is equivalent to setting up the memory data dependence problem. Then enter that into the barnivok/iscc calculator (http://www.cs.kuleuven.be/cgi-bin/dtai/barvinok.cgi) and also show the iscc calculator output.

(a)
for(int i=0; i<N; ++i)
    for(int j=0; j<i; ++j)
        for(int k=i; k<N; ++k)

(b)
for(int i=0; i<N; ++i)
    for(int j=i; j<N; ++j)
        A[j] = A[2i-1];

(c)
for(int i=0; i<42; ++i)
    x[i+42] = 2*x[i]

4. [5 points] Exact data dependence analysis
For the following loop, set up the EXACT data dependence analysis problem for the flow data dependence. Show the initial set of constraints and then show the simplified data dependence relation from iscc.

for(int i=0; i<N; ++i)
    for(int j=i; j<N; ++j)
        A[i] = A[i-1];

5. [10 points] Iteration Space in Matrix Format
for (i=2; i<N; i++) {
  for (j=i+1; j<N-2; j++) {
    A[i][j] = ...  
    ... A[i-2][j+2] ...
  }
}

(a) For the above loop, express the iteration space as a polyhedral set using a one matrix format.
(b) Also show how to express the set to iscc and the iscc output.
(b) Use iscc to generate code for this original iteration space.

6. [10 points] Applying the transformation to the iteration space

(a) Apply the transformation $T_{I \rightarrow I'} = \{ [i, j] \rightarrow [i', j'] | i' = j \land j' = i + j \}$ to the iteration space from problem 1 and show the resulting one matrix format for the new iteration space $I'$. Note that the one matrix format for the transformation is as follows:

$T_{I \rightarrow I'} = \{ [i, j] \rightarrow [i', j'] | \begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \}$

(b) Use iscc to generate code for the transformed iteration space.

7. [10 points] Array accesses

(a) For the loop from problem 1, show each array access in a one matrix format.
(b) Also, show how the transformation $T_{I \rightarrow I'}$ transforms the array accesses, and show the new array accesses in a one matrix format.
(c) Does this match the actual parameters passed to the statement macros in the iscc generated code?

8. [10 points] Data dependence relation

(a) For the above loop, show the data dependence relation using the one matrix format.
(b) Additionally, show how the transformation $T_{I \rightarrow I'}$ transforms the data dependence relation, and show the new data dependence relation in a one matrix format.