Preliminaries:
1) Submission:
Send me by email to pouchet@colostate.edu, subject "CS560: assignment 2 -- your name" a text file containing the full ISCC code to
compute/display the answer to all questions above. I must be able to copy-paste as-is your text file into the online ISCC calculator, and run without error.
The hard deadline is Tuesday, October 17 2017, 11:59pm MT. No extension will be provided.

2) Program. This assignment focuses on studying this program, which smoothes an image TimeSteps time. It is also the prototype code,
in terms of dependences, for iterative Jacobi solving of partial differential equations. We call this program Jacobi-2D.

```
for (t = 0; t < TimeSteps; ++t) {
  for (i1 = 1; i1 < N-1; ++i1)
    for (j1 = 1; j1 < N-1; ++j1)
R:    B[i1][j1] = 0.2 * (A[i1][j1-1] + A[i1][j1] + A[i1][j1+1] + A[i1+1][j1] + A[i1-1][j1]);
  for (i2 = 1; i2 < N-1; ++i2)
    for (j2 = 1; j2 < N-1; ++j2)
S:    A[i2][j2] = 0.2 * (B[i2][j2-1] + B[i2][j2] + B[i2][j2+1] + B[i2+1][j2] + B[i2-1][j2]);
}
```

3) ISCC. This assignment uses the ISCC calculator. An online version is available at
http://compsys-tools.ens-lyon.fr/iscc
and you can download/install locally Barvinok, which contains ISCC, from
http://barvinok.gforge.inria.fr
Note Linux may be needed to install this tool, a lab machine will work.

4) Questions. Questions below are of the form "write xx", "compute yy", "count zz", etc. For each, you must provide the ISCC command(s)
such that, when executing your script in ISCC, "xx", etc. is being printed out. For example, if the question is "write the iteration domain of xx",
then a possible answer is as follows:

```
IterDom := { [i,j] : 0 <= i,j < 42 };
IterDom;
```

Which will produce the ISCC output:

```
{ [i, j] : 0 <= i <= 41 and 0 <= j <= 41 }
```

that is, which will print the iteration domain. If the question is "count the number of points in xx", a possible answer is:

```
IterDom := { [i,j] : 0 <= i,j < 42 };
NbPoints := card IterDom;
NbPoints;
```

Which will produce the ISCC output:

```
{ 1764 }
```

Part 1: Representing programs.

Question 1: Write the iteration domain of R, and of S. Count the number of points in each.

Question 2: Write each access function (12 total) for the program.

Question 3: Write the union of all access functions on the array A.

Question 4: Write the union of all access functions on the array B.

Part 2: Analyzing programs.

Question 5: Write the data space of A, i.e., the set describing all cells of the array A being referenced over the execution of the program.

Question 6: Write the data space of B.

Question 7: Count the number of points in the data space of A, and B.

Question 8: Count the sum for R and S of the number of executed statement instances (hint: make the union of iteration domains first, then count,
as ISCC does not provide arithmetic operations such as +, *, etc. on integers)

Question 9: Count how many times each data element of A is being accessed by the program. Note: this is a more difficult question, not covered in class.

Part 3: Generating communication code.

Question 10: Compute the data space DS_R_A_j1 of A for one arbitrary execution of the j1 loop, that is the parametric slice fixing t, i1, i2, j2 to a newly
introduced parameter. Don't forget to update all your data structures -- iteration domains and access functions -- to capture the newly introduced parameters.

Question 11: Generate the code scanning DS_R_A_j1.
We will now implement local memory promotion in the program. The shape of the final code is:

```
for (t = 0; t < TimeSteps; ++t) {
  for (i1 = 1; i1 < N-1; ++i1) {
    float A_local[??][??];
    float B_local[??][??];
    // code to copy-in data from A[][] to A_local[][]
    ??
    for (j1 = 1; j1 < N-1; ++j1)
R:    B_local[??][??] = 0.2 * (A_local[??][??-1] + A_local[??][??] + A_local[??][??+1] + A_local[??+1][??] +
A_local[??-1][??]);
    // code to copy-out data from B_local[][] to B[][]
    ??
    }
  for (i2 = 1; i2 < N-1; ++i2)
    for (j2 = 1; j2 < N-1; ++j2)
S:    A[i2][j2] = 0.2 * (B[i2][j2-1] + B[i2][j2] + B[i2][j2+1] + B[i2+1][j2] + B[i2-1][j2]);
}
```

Where we use local arrays (A_local and B_local), akin to using shared memory on GPUs. The objective is to use

ISCC to automatically generate the program that uses local memory,
that is replacing ?? everywhere by ISCC-generated code/values such as the size of the local arrays needed and the code to copy-in and copy-out the data to/from the local arrays,
akin to copying from/to main memory on GPUs. It is about putting together the pieces computed in this assignment so far.

Question 12: Count DS_R_A_j1. Compute the size of A_local in each dimension (observation: you should compute the size based on the rectangular hull of DS_R_A_j1).

Question 13: Compute DS_R_B_j1 (same as DS_R_A_j1 but for the B array), and compute the size of B_local as above.

Question 14: Generate the loop code that copies data from A[x][y] to A_local[a][b] in a correct manner, i.e., not exceeding the size of A_local but still copying
exactly the data needed for this particular execution of the j1 loop. Hint 1: this is very related to question 11. Hint 2: the code for the statement S(i,j) you will
use is up to you, e.g., it could be:

print "A_local[i-I][j-J] = A[i][j];";
where I and J are some parameters. As ISCC does not generate C-compilable code as-is, to answer this question you should (1) generate the loop code (codegen) and
(2) print the prototype statement body you use, as above.

Question 15: Generate the loop code that copies data from B_local[x][y] to B[a][b] in a correct manner.

Question 16: print the final statement body for R to be used (where all ?? are replaced by their correct value).

Question 17 (optional, difficult): Compute the data reuse between two consecutive executions of the j1 loop. Generate the loop code that copies data from A[x][y] to A_local[a][b]
(only one row of A each time now), and print the final statement body for R (hint: % can be used in the array accesses).

Thanks,

++