CS243 Winter 21, Section 6

Problem 1. Affine Transforms.

Consider the following program.

```
for (int i = 1; i < n; i++) {
  for (int j = 0; j < n; j++) {
     A[i, j] = c * A[i, j-1];
  }
}
for (int i = 1; i < n; i++) {
  for (int j = 1; j < n; j++) {
    B[i, j] = (B[i-1, j] + A[i, j] + A[i-1, j]) / 3;
  }
}
```

Assume `A` and `B` are two non-overlapping $n \times n$ matrices. Both matrices are stored in row-major layout.

1. Draw the iteration space for the program. Use arrows to mark data-dependencies between iterations.

2. Parallelize each loop nest individually. Show the transformed code, and describe any transformation you performed.

Problem 1. Affine Transforms (cont.)

(Same program copied here for reference.)

```
for (int i = 1; i < n; i++) {
  for (int j = 0; j < n; j++) {
     A[i, j] = c * A[i, j-1];
  }
}
for (int i = 1; i < n; i++) {
  for (int j = 1; j < n; j++) {
    B[i, j] = (B[i-1, j] + A[i, j] + A[i-1, j]) / 3;
  }
}
```

3. Can you parallelize this program with no synchronization? Show the transformed code, or provide a justification that it is not possible.

Problem 1. Affine Transforms (cont.)

(Same program copied here for reference.)

```
for (int i = 1; i < n; i++) {
  for (int j = 0; j < n; j++) {
     A[i, j] = c * A[i, j-1];
  }
}
for (int i = 1; i < n; i++) {
  for (int j = 1; j < n; j++) {
    B[i, j] = (B[i-1, j] + A[i, j] + A[i-1, j]) / 3;
  }
}
```

4. Can this program be written as a single loop nest with pipelined parallelism? Show the fully permutable loop nest and the best generated parallel code (using any necessary synchronization primitive, as in Homework 6), or provide a justification that it is not. You do not need to perform blocking.

Problem 1. Affine Transforms (cont.)

(Same program copied here for reference.)

```
for (int i = 1; i < n; i++) {
  for (int j = 0; j < n; j++) {
     A[i, j] = c * A[i, j-1];
  }
}
for (int i = 1; i < n; i++) {
  for (int j = 1; j < n; j++) {
    B[i, j] = (B[i-1, j] + A[i, j] + A[i-1, j]) / 3;
  }
}
```

5. Which of the original code, or any of the parallel versions, is likely to perform best? Assume any necessary blocking is now performed.