The purpose of this lab is to familiarize you with the basics of DMA and using SPE threads. The lab is short and shouldn't take more than 30 to 60 minutes.

The lab is due before the next recitation (10am Tuesday, January 16). Code should be written individually, though feel free to collaborate and work with others.

The problem:

Given 2 lists of points in a 2D coordinate system, calculate the pairwise distance between them. Coordinates are single-precision floating point numbers (4 bytes each) and each list contains exactly 64 points.

The given code contains the following:

```
#define NUM_POINTS 64
#define CACHE_ALIGNED __attribute__((aligned(128)));

POINT a[NUM_POINTS] CACHE_ALIGNED;
POINT b[NUM_POINTS] CACHE_ALIGNED;

float dist[NUM_POINTS][NUM_POINTS] CACHE_ALIGNED;
```

`a[]` and `b[]` are the 2 lists of input points; we want to fill in `dist[][]` so that

> `dist[i][j]` contains the distance between `a[i]` and `b[j]`

You should use 2 SPEs to perform the computation. Note that `sizeof(a)` is 512 bytes and `sizeof(dist)` is 16 KB, so everything can be transferred in one DMA command. Don't worry about floating-point precision or overflows. Your code also doesn't need to be efficient or use intrinsics.

The given code in `6181lab1.zip` contains:

- `common.h`: definitions; you may need to add more
- `dist.c`: a test framework and PPU implementation. Replace the `calc_dist()` function to use 2 SPEs
- `spu/`: a Makefile that you can use for your SPU program

The given PPU code generates random points for arrays `a[]` and `b[]`, calls `calc_dist()`, and checks that the distances calculated are correct. `calc_dist()` currently does the computation on the PPU. You need to modify it to instead use 2 SPEs.

Hints:

The `dist[][]` array is laid out linearly in memory; i.e., as:

```
{dist[0][0], dist[0][1], ... dist[0][63], dist[1][0], dist[1][1], ...}
```

As a result, you can use each of the 2 SPEs to compute half of the `dist[][]` array:

- Divide list `a[]` in half; send half to one SPE, and the other half to a second SPE
- Send list `b[]` to both SPEs
- Have each SPE calculate pairwise distances and DMA results back to different halves of the `dist[][]` array in memory

Use the `distf(POINT *, POINT *)` function to compute the distance between 2 points. Alternatively, you can use the `sqrt(double)` function to compute square roots and the `sqrf(float)` function to compute squares. These are declared in `common.h`.

More hints:

Use the approach in the DMA example from recitation. You'll need a different control block for each SPE that contains:

- The address of the part of list `a[]` the SPE is processing
- The address of list `b[]`
- The address of the part of `dist[][]` the SPE will copy output to

Caveats:

Don't use the `sqrtf(float)` function declared in `math.h`. This seems to be inaccurate. `sqrt(double)` works fine.