```
for (int i = 0; i<max; i++){
 float tmpA = 0.0;
 for (int j = 0; j<max2; j++){
    tmpA += 2*B[j];
    ...
    }
 ...
}
```

Figure 1: A code snippet with tmpA initialized for every iteration.

```
float tmpA[max] = {0.0, ...};
for (int i = 0; i<max; i++){
 for (int j = 0; j<max2; j++){
    tmpA[i] += 2*B[j];
    ...
    }
 ...
}
```

Figure 2: The same code with tmpA hoisted.

0.1 CPU Parallelization Preparation

The first step for us was to begin preparing the CPU for kernel parallelization. Before distributing the outer loops in ProjCoreOrig.cpp, we began by moving all of the secondary functions (`void updateParams`, `void setPayoff`, `REAL value`, and `void rollback`)into `int main`. This allowed us to easily see the globs (global variables) and their relations to one another.

Conglomerating the separate functions into one allowed us to hoist the initialization of the globs into glob arrays. That is, instead of initializing a temp variable or array for every iteration of a loop, we instead initialize an array one dimension larger, with size equal to the number of iterations of the loop, before the looping code. See figures 1 and 2 for an example.

The purpose of this is twofold: one, we can in reduce the number of times the initialization computation is performed; and two,

0.1.1 section notes while writing

the initialization of the glob array has been hoisted out prev, every iteration had an init of globs, now we have an array of globs and it is initialized beforehand, in each init it was init to same val first create init, create array where defaullt is the init value.

0.2 Loop

0.3 Simple CUDA

0.4 TRIDAG