COSC175 Day23: Exploring Locality

Start by **assigning roles** in your group! If you have only 3 members, the Ambassador+Recorder roles are combineable, and for groups of 5, multiple Ambassadors or Sketchers will help!

- ***Ambassador*** should have the textbook, lecture materials, and other examples handy, and is the point-person for raising their hand to ask a question.
- ***Recorder*** should take detailed notes, either on the handout or in a digital format, and will submit notes on behalf of the group at the end of class.
- ***Sketcher*** should have a marker in hand, drawing at the board as your group brainstorms and works through the exercises.
- ***Executive*** should keep an eye out that all group members are staying in-the-loop, as well as an eye on the clock in case you are stuck on a point, would benefit from help, or would benefit from moving on to another question.

With your group **at the whiteboard**, work through the following exercises to brainstorm about locality in programs!

You may want to remind yourself of the 2D array memory layout examples discussed here, and to have a picture of the program memory address space (e.g., DDCA Figure 6.31) on hand as a reminder.

1. Describe the spatial and temporal locality exhibited in Example A below;
    1. How would you categorize the access behavior of each variable (i.e., does use of `sum` exhibit temporal or spatial locality? What about `a` ?)?
    2. How would you describe the locality within each segment of the address space (i.e., Text (the instructions) vs. Stack)?
    3. If you are the compiler (translating from C down into assembly), what optimizations might you like to make to reduce the number of RAM accesses? For example, which commonly-used variables might you assign to registers?

2. At the programming level, what would be the impact of changing the order of the loops, – in other words, iterating over i, then j, versus iterating over j, then i? (Hint: consider large values of M or N and the number of pages in use during 1 iteration of the outer loop.)
3. (Bonus, only if time) repeat 1, 2, 3 for Example B

Example A:

```
// a simple function in C that sums the values in a 2D array of int
int sumarray(int a[M][N]){
  int i, j, sum = 0;
  for (i = 0; i < M; i++){
    for (j = 0; j < N; j++){
      sum += a[i][j];
    }
  }
  return sum;
}
```

Example B:

```
// a snippet of C code that multiplies two matrices
int n = 10; // dimension of # rows and columns for each matrix
int a[n][n];
int b[n][n];
int output[n][n];

// initialize values in a and b, and initialize output to all zeros
for (int i = 0; i < n; i++){
  for (int j = 0; j< n; j++){
    a[i][j] = i + j;
    b[i][j] = j;
    output[i][j] = 0;
  }
}

// accumulate output based on a and b
for (int i = 0; i < n; i++){
  for (int j = 0; j < n; j++){
    for (int k = 0; k < n; k++){
      output[i][j] += a[i][k] * b[k][j];
    }
  }
}
```

Before the end of class be sure your recorder hands in a copy of your notes with all group members listed! You can hand in notes on paper or via email with [COSC175] in the subject line. I use these notes to check your understanding and your engagement, not to grade for correctness.