**CS520—Spring 2013—Homework 11**
**Wednesday April 17**

### Question 1

Label the following code to indicate where spatial and temporal locality is exhibited.

```
#define INTERVALS 50000000

main (argc, argv)
   int argc; char *argv[];
{
   int i;
   double sum;       /* Sum of areas */
   double width;     /* Width of interval */
   double x;         /* Midpoint of rectangle on x axis */

   start_timer();

   width = 1.0 / INTERVALS;

   sum = 0.0;
   x = width * .5;
   for (i = 0; i < INTERVALS; i++) {
      sum += 4.0/(1.0+x*x);
      x += width;
   }
   sum *= width;

   stop_timer();

   printf ("Estimation of pi is %14.12f\n", sum);
}
```

### Question 2

Understanding how two-dimensionals arrays are laid out in memory is important for understanding memory cache behavior. Do experiments to determine how C two-dimensional arrays are laid out. Are they row-major (row elements are adjacent in memory) or are they column-major (column elements are adjacent in memory)? Describe your experiments that helped you answer this question.

### Question 3

Consider how the following C loop would be accessed by a direct-mapped cache with 8 sets and a block size of 4 ints:

```
int i, j, sum, a[16][16];

sum = 0;
for (i = 0; i < 16; i++)
  for (j = 0; j < 16; i++)
    sum += a[i][j];
```

Assume only the array is in the cache and the first word of the array is at address zero. If the cache is initially empty, how many cache hits would there be?

**Question 4**

Re-do Question 2 except swap the array indices in the body of the loop. That is, analyze this code:

```
int i, j, sum, a[16][16];

sum = 0;
for (i = 0; i < 16; i++)
  for (j = 0; j < 16; i++)
    sum += a[j][i];
```

How many cache hits would there be in this case?