CS/ECE 560 Foundations of Fine-Grained Parallelism — Spring 2012	**HW #6**
AlphaZ Exercise	**Due April 4th, 2012 at 11:59pm**

ALL of the homeworks and project writeups in this course must be written using latex. See the template latex file (cs560-template.tex) at http://www.cs.colostate.edu/~cs560/Spring2012/assignments.php. Homework assignments are to be completed individually. Homeworks need to be submitted electronically via RamCT AND by email to cs560@cs.colostate.edu by 11:59pm on the due date. For this homework submit the pdf that you built from latex. Total points: 100

1. [40 points] Reductions: Maximum Segment Scan(s)
   Given an array of N values. A segment in the array is any sequence, $[A_i ... A_j]$ for $0 \leq i \leq j < N$ (so there are about $n^2/2$ segments). A segment sum $S[i,j]$ is the sum of all the elements in the segment $\sum_{k=i}^{j} A_k$. The maximum segment sum m is the maximum of $S[i,j]$ over all segments, and it can be computed with the following equation

   $$m = \max_{0 \leq i \leq j < N} \sum_{k=i}^{j} A_k.$$

   Write an Alphabets program to compute the Maximum Segment equation. Verify your program by specifying a serial schedule (see http://www.cs.colostate.edu/AlphaZ/wiki/doku.php?id=target_mapping) and generating and running the corresponding C program on the following example sequence $[-1, 3, -5, 7, -6, 4, 2]$.

2. [60 points] C code to Alphabets
   In this problem you are asked to convert the given C code for a Jacobi 2D stencil computation into an Alphabets program.

```
// Initialize a[i][j] and b[i][j] with the value j-i.
for (t=0; t<T; t++) {
    for (i=1; i<N-1; i++) {
        for (j=1; j<N-1; j++) {
            b[i][j]= 0.2*(a[i][j]+a[i][j-1]+a[i][1+j]+a[1+i][j]+a[i-1][j]);
        }
    }
    for (i=1; i<N-1; i++) {
        for (j=1; j<N-1; j++)   {
            a[i][j]=0.2*(b[i][j]+b[i][j-1]+b[i][1+j]+b[1+i][j]+b[i-1][j]);
        }
    }
}
```

   (a) Write the alphabet program for that then choose a schedule and processor allocation for it with at least one parallel loop.

(b) Use the AlphaZ verifier to show that the schedule and processor allocation is legal.

(c) Generate scheduled code for this program and verify that the generated program produces the the same answers as the original C version.

(d) Finally, play with the storage mapping in AlphaZ to determine how to have the generated code use the same amount of storage as the original C code. (Ask questions about this one!)