1. (30 points) Consider the following computation:

```
for t = 1,nt
    for i = 1,N
        for j = i,N
            A(i,j) = (A(i-1,j)+A(i,j-1)+A(i+1,j)+A(i,j+1))/4
        end for
    end for
end do
```

(a) List all data dependences, stating the kind of dependence and the distance vector.
(b) What permutations are valid? Why?
(c) Which loops are valid to unroll? Why?
(d) Is 3-dimensional tiling valid? Why?

2. (40 points) Consider the following 2D and 3D forms of tiling for matrix-matrix product.

```
for it = 1,1024,512 for it = 1,1024,256
    for jt = 1,4096,512 for jt = 1,4096,256
        for i = it,it+511 for kt = 1,4096,256
            for j = jt,jt+511 for i = it,it+255
                for k = 1,4096 for j = jt,jt+255
                    C(i,j) = C(i,j) + A(i,k)*B(k,j)
                end for
            end for
        end for
    end for
end for
```

(a) Assuming a fully associative cache with linesize 8 words and capacity of 512 Kwords, which of the above tiled forms is preferable? Assume row-major storage of arrays.
(b) Would tiling two other dimensions (instead of i and j) be better for the 2D tiled version? If so, how much better? State any assumptions in your calculation.
3. (30 points) A dot-product version (ij form) of code for solving a lower-triangular system of equations $Ax = b$ is shown. The coefficient matrix for the lower-triangular system is held in $A$ ($A$ is a full square matrix; but almost half its elements are unused). Initially, the r.h.s. vector is held in $b$. At the end of the computation, $x$ holds the solution vector. The r.h.s. vector $b$ is modified by the code. When row $i$ is processed, $a_{ij} * x_j$ is subtracted from $b_i$, for all $j$ less than $i$ (the value of $x_j$ has been already finalized for those $j$’s. The structure of the code is quite similar to that for matrix-vector multiplication, but there are differences in the data dependences.

```c
for i = 1,N
    temp = b(i)
    for j = 1,i-1
        temp = temp - a(i,j)*x(j)
    end for
    x(i) = temp/a(i,i)
end for
```

(a) Create a version with a two-way unrolled outer-loop (assume N is even).
(b) Create a SAXPY form (ji form) for the lower-triangular solver.
(c) Assuming that matrices are stored in row-major order, which form would you expect to have better performance?
(d) Compare the performance of the different versions on stdsun, for $N = 512$. 