Exercise 3

Solution 3.1 (Exam 2007: OpenMP)**.** Develop parallel versions of the following codes by augmenting them with OpenMP directives. Try to reduce the number of implicit barriers and specify every used variable as `shared` or `private`.

a)
```
#pragma omp parallel for private(i,j) shared(y,A,x,m,n)
    for (i=0; i<m; i++){
        y[i]=0;
        for (j=0; j<n; j++){
            y[i]= y[i] + A[i][j]*x[j];
        }
    }
```

b)
```
s=0;
#pragma omp parallel for private(i,j) shared(y,A,x,m,n) reduction(+:s)
for (i=0; i<m; i++){
    y[i]=0;
    for (j=0; j<n; j++){
        y[i]= y[i] + A[i][j]*x[j];
    }
// or, instead of reduction:
// #pragma omp critical
    s += y[i];
}
```

c) Without adding additional variables this code can not be parallelized with OpenMP. However if one add a single variable it is solvable:

```
    y[0]=0;
    for (i=1; i<m; i++){
        s = 0;
#pragma omp parallel for private(j) shared(A,x,n,i) reduction(+:s)
        for (j=0; j<n; j++){
            s += A[i][j]*x[j];
        }
        y[i] = s + y[i-1];
    }
```

d) Parallelize the following three loops in a single parallel region and use the `nowait` clause whenever possible.

```
#pragma omp parallel private(i,j) shared(A,y,m,n,x,alpha)
{ // open a block
```

```
#pragma omp for nowait
    for (i=0; i<m; i++){
         y[i]=0;
         for (j=0; j<n; j++){
             y[i]= y[i] + A[i][j]*x[j];
         }
     }
#pragma omp for
     for (i=0; i<m; ++i){
         alpha[i]=A[i][0]*A[i][1];
     }
#pragma omp for
     for (i=0; i<m; i++){
         y[i] = y[i]*alpha[i];
     }
} // close block (parallel region)
```

Solution 3.2. As discussed in the class, there are differences between compilers, in how they handle this code. The bottomline is that, while taskwait syncronizes child tasks only, taskbarrier synchronizes threads, and, therefore, their associated tasks.