OpenMP Assignment

Instructions: Q1 is mandatory. You have to do one of the other questions; you can choose to do either Q2 or Q3. ALL THE CODES SHOULD BE COMPILED WITH gcc only.

Q1. Write a task based OpenMP program in C that implements the dynamic program to solve the subset sum problem with memoization using recursion. Assume inputs are non-negative integers only and n=number of inputs and S=total desired sum are provided as input. Assume that the function is invoked from a parallel region (i.e., you should not have the #pragma omp parallel construct inside your function).

See https://www.geeksforgeeks.org/subset-sum-problem-dp-25/ for (i) a recursive implementation as well as a (ii) dynamic program without memoization. See https://en.wikipedia.org/wiki/Memoization for more details on memoization.

Q2. Define a suitable data structure my_struct and write routines mylock_init(my_struct), mylock_lock(my_struct), mylock_unlock(my_struct) and mylock_destroy(my_struct) to implement locks. You are only allowed to use #pragma omp atomic to implement the lock.

See https://scc.ustc.edu.cn/zlsc/tc4600/intel/2016.0.109/compiler_c/common/core/GUID-630D512F-C1A3-4F92-81D3-D2457EDBD572.htm to understand what you can and cannot achieve using atomic.

You should test your implementation by initializing a large array with 1s and summing up the elements using multiple OpenMP threads with a lock around the update of the shared variable sum.

Q3. Consider the following code for matrix transpose

```
for ( i = 0 ; i < n ; i++ )
      for ( j = 0 ; j < i ; j++ )
            swap( a[i][j], a[j][i] )
```

This code traverses the i-th row (upto diagonal) and i-th column (upto diagonal) and swaps the elements. This code is inefficient as row access is cache-efficient but the column access is not. There is a block representation of matrices (see https://en.wikipedia.org/wiki/Block_matrix) in which the matrix is divided into sub-blocks. Consider the code that transposes pair of blocks across the diagonal instead of elements. If the block is small enough (the elements of a row of the block cover the cache-width), then the code is cache-efficient along the rows and columns of the blocks.

Write an OpenMP program to perform a blocked matrix transpose.