CS376 Lab 5 Fall 2016
TSP Problem
Due Tuesday 10/25/2016 and Tuesday 11/8/2016 at 11:59PM

Based on the `permute.cpp` example, create a sequential program named `tspseq.cpp` to solve the traveling salesperson problem (TSP). The only way to get the optimal solution is to check every path and check the shortest one. The permute program allows you to generate all the permutations (there are n! of them). Instead of printing them out, calculate the length of the path and update a global variable with the shortest path. Since the path 0, 1, 2, 3 will have the same length as path 2, 3, 0, 1, we will always start the path at city 0. Because of this you will want to generate the permutations of 1, 2, 3 (not 0, 1, 2, 3) and then calculate the path length assuming you start at city 0 and end at city zero (i.e., for the permutation 1, 2, 3, the path length is the sum of the paths from `0->1, 1->2, 2->3, and 3->0`). The following is a sample input file.

```
4
0 6 5 5
5 0 9 6
7 6 0 8
3 5 3 0
```

The first line holds the number of cities (and specifies the number of remaining lines in the file). The remaining lines specify the distance from each city to each city. For example, the line 0, 6, 5, 5 indicates the distance from `0->1` is 6, `0->2` is 5, and `0->3` is 5. The next line indicates the distances from city 1 and so on.

Your sequential program must produce output formatted exactly the same as below

```
sequential time for 4 : 0
19
0 3 2 1
```

The first line indicates the number of cities and the time it took to calculate the path. The second line is the path length. The third line is the order of the cities to visit (i.e., in this case, you go from 0 to 3 to 2 to 1 to 0).

For the parallel version change the word “sequential” in your output to either “pthreads”, or “gcd” to indicate which parallelization method you used.

Assume we will never call the code with more than 20 cities and use a constant size of 20 for any arrays you use.

The sequential version is due Tuesday 10/25 at 11:59PM and will be graded in the homework category.

By Tuesday 11/8 at 11:59PM, submit a parallel version. For full credit your program must achieve a speedup over the sequential version for input sizes of 13 and 14 cities. You may use pthreads or Grand Central Dispatch for your parallel program. Name your file `tsppar.cpp`.

You may use any technique you want to parallelize the code as long as it achieves a correct answer. Note there can be more than one path with the same minimum length. My recommended algorithm is to make the first set of recursive calls in your `main` function (i.e., use a loop to call your `permute` function that matches the first set of recursive calls made in the permute function). Think about the various `count3` examples so that not all your code is updating the same global path or you will not get a speed up due to the penalty of locking a global variable every time.

When you are finished, submit your lab by emailing your `tspseq.cpp` file for the homework grade and `tsppar.cpp` for the lab grade as an attachment from your Capital email account to `dreed@capital.edu` with the subject `CS376ATT` (note there are no spaces and it is all capital letters).

I typically do not notice any comments in the body of the message as these emails are filtered automatically. Please send a separate email with a different subject if you have a question.

Your lab will be graded using the rubric posted on iLearn.

If you realize you made a mistake, you may resubmit your program (before the due date). I only keep and grade the last submission. When resubmitting, send all the specified files.

If you use GCD, you cannot use arrays inside of blocks. This is due to the change in the C specification that allows arrays to have a dynamic length. Instead you will need to create a class that contains an array with a fixed size. For example, any time you want to access an array inside a block, you will need to do something such as:

```
const int SIZE = 20;
class Array {
public:
    int items[SIZE];
};
```

and then use an instance of Array. So instead of writing:

```
const int SIZE = 20;
int a[SIZE];
a[0] = 0;
```

you need to write:

```
Array a;
a.items[0] = 0;
```

Variables of type `Array` can be accessed inside the block.

Please note that if your code does not implement a parallelization method for the parallel version, you will receive a zero (i.e., submitting the sequential version for the parallel version will result in a zero for the lab grade).