1) Amdahl

You are given the following code.

```
int my_function( int n )
   {
   double *M;                  // M is an array
   M = (double *) malloc( n * sizeof(double) );
   double res;                 // result

   if( my_test( n ) )
      init_A( M, n );
   else
      init_B( M, n );

   res = process( M, n );

   post_process( res );

   return 0;
   }
```

Independently of `n`, the following relations hold.

- T_1(`init_A`) $= \frac{1}{10}\ T_1$(`my_function`)
- T_1(`init_B`) $= \frac{1}{10}\ T_1$(`my_function`)
- T_1(`process`) $= \frac{17}{20}\ T_1$(`my_function`)
- T_1(`post_process`) $= \frac{1}{20}\ T_1$(`my_function`)

The functions `init_A` and `init_B` are strictly sequential.

Answer the following questions.

a) When parallelizing `my_function`, what is the maximum achievable speedup?

b) What is the maximum achievable speedup for `my_function`,
if only the function `process` is parallelized?

2) Weak Scalability

You want to use the program `TotalSimulator` to solve a large problem on a supercomputer. Before submitting your job, you collect data to estimate the execution time of your simulation. You acquire the following information.

- As input, `TotalSimulator` takes an integer n.
  You know that the complexity of the program is quadratic in n, that is $O(n^2)$.
  You also know that program operates on $O(n)$ data.
- For $n = 50{,}000$, `TotalSimulator` achieves a nearly perfect efficiency for up to 256 processors. When executed with $p > 256$ processors, you expect a 10% loss of efficiency for each doubling of p. (Thus, for instance, the efficiency for 1024 processors is expected to be 80%.)
- $T_{128}(50{,}000) = 12.3$ secs.

Answer the following questions.

a) Assuming that the efficiency is sustained with respect to the problem size, what is $T_{2048}(200{,}000)$?

b) Assuming again that the efficiency is independent from the problem size, use the observed $T_{128}(50{,}000) = 12.3$ secs as a reference, and fill out the following table, to estimate the weak scalability of `TotalSimulator` with p in the range $[32, \ldots, 2048]$.

p	32	128	256	512	1024	2048
Size		50k				
Time		12.3				