SABANCI UNIVERSITY

CS406/531: Parallel Computing
Spring 2018-2019, Midterm I
Duration: 110 minutes
Grade: / 100

Q1 (15pts): Suppose a fraction **r** of the runtime of a serial program is "*perfectly parallelized*," and the remaining fraction **1 − r** is "*inherently serial*." Give an upper bound on the speedup one can have.

Q2 (15 pts): Describe the following terms and their differences in at most four sentences each.

a) (5 pts) Latency **and** bandwidth
b) (5 pts) Spatial locality **and** temporal locality
c) (5 pts) Shared address space **and** distributed address space

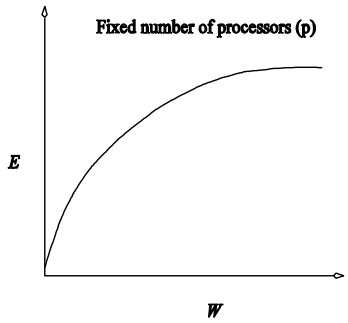
Q3 (10 pts): Implement a parallel function with OpenMP to reverse an array in five lines. Try to be as efficient as possible. The function **must be not more than 5 lines and be an in-place function**, i.e., using extra memory is not allowed.

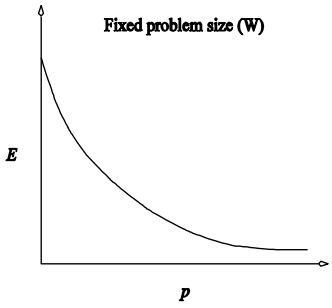
```
#include <iostream>
#include <memory>

using namespace std;

//5 lines including pragma – no memory allocation
void reverse(int* arr, int length) {









}

int main() {
  int N = 1000000;
  int* arr = new int[N];
  for(int i = 0; i < N; i++) {
    arr[i] = rand();
  }
  reverse(arr, N);
}
```

Q4 (10pts): Answer the following questions about the charts below: (**E** is efficiency, **p** is number of processors, **W** is the problem size).

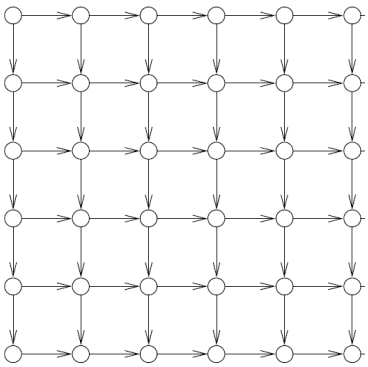
a) Which chart (left or right) is correct for **all** parallel systems (architecture + algorithm)? Why?
b) Which chart (left or right) is correct for **some** parallel systems (architecture + algorithm)? What do we call such systems and why?

Answers without correct explanations won't get partial credit.

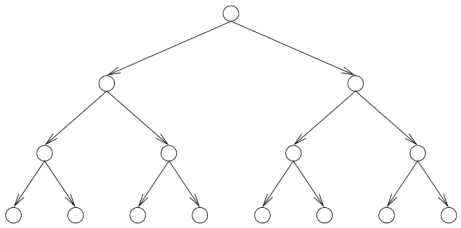




Q5 (15pts): Given two sample task dependency graphs as in the figures (5.1) and (5.2) of **N uniform** tasks. **Assume the graphs below are only instances of a family of task dependency graphs following the same pattern**, so **n** and **N** can be different.



(5.1) N = n^2



(5.2) N = $2^n - 1$

a. (5pts) What is the critical path length for each graph **in terms of n and N**?
b. (5pts) What is the degree of concurrency for each graph **in terms of n and N**?
c. (5pts) What is the speedup and efficiency one can obtain **in terms of n and N** if the number of processing elements is equal to the degree of concurrency? Show all your work.

Q6 (15pts): Consider a processor operating in 4GHz connected to a DRAM with a latency of 25ns (no caches). Assume the processor has two multiply add units and is capable of executing four instructions at each cycle of 1ns.

a. (3pts) What is the peak performance of the processor in FLOPS?
b. (5pts) Consider the problem of computing the dot product of two vectors (for one multiply-add, we need two data items): What is the peak performance in FLOPS for that algorithm? Explain the math you use.
c. (3pts) Assume that you have a single cycle cache and the block size (memory bandwith) is one word. What is the peak performance for dot product computation in this case?
d. (4pts) What is the peak performance again with a single cycle cache if the block size is 8?

Q7 (10 pts): Suppose that you implemented a function **party()** as shown below:

```
int party() {
  int a = 0;
#pragma omp parallel num_threads(3)
  {
    for(int i = 0; i < 5; i++) {
      a = a + 1;
    }
  }
  return a;
}
```

a) (3 pts) While building the executable, you forgot to use **–fopenmp**. The code still compiles; what are the maximum and minimum values the function party can return?
b) (7 pts) You added **–fopenmp** to build the executable. What are the maximum and minimum possible values the function can return?

Q8 (10 pts): Please answer the following:

a. (5 pts) Given a sequential algorithm requiring **O(n)** operations on **n** numbers, assume that you devised a parallel version requiring **O(n/p + log p)** time with **p** processors. What is the efficiency (E) of your algorithm?
b. (5 pts) (Continue from part (a)): When (up to how many processors) is your algorithm cost optimal? Show all your math, only the answer will not get partial credits.