Preliminary Examination
Department of Scientific Computing

Parallel Programming

1. A character array **s** contains the text of a book, and consists of exactly 10,000,000 characters. The only chararacters are lowercase alphabetic characters ('a' through 'z') and blanks. We may regard the text as a sequence of words of various lengths, where words are strings of nonblanks separated by one or more blanks.

a Describe an algorithm to count the number of words in **s**;
Answer: *Counting words is the same as counting the characters that begin words. A character begins a word if it is not a blank, and either it is the first word in the text, or the preceding character is a blank. Therefore, to count words, we can simply examine each character, and increment the counter if*

- *it is the first character and not a blank, or*
- *if it is a subsequent character, not a blank, and the preceding character was a blank.*

b We wish to use parallel processing to count the number of words in **s**. We are going to try to do parallel processing without using MPI. Suppose we have 10 separate computers, we cut the text into 10 parts, we run the same program on each computer, and we take the partial sums from each computer and add them up by hand. Even though the computation might be faster, the result will probably be slightly incorrect. Why is the sum of the partial word counts probably not the exact number of words in the text?
Answer:*The first process has enough information to correctly identify characters that begin words. Subsequent processes do not. If the first character that the second process sees is a nonblank, then it might be the beginning of a word, or the continuation of a word. Thus, every process except the first could incorrectly count one extra word.*

c Now we want to use MPI to carry out the task. Again, we plan to use 10 processes, and each process will only have access to 1/10 of the data. The fact that the MPI processes can communicate is very important, and

means that we can avoid the error noticed in the previous section. Focus on a single MPI process with identifier ID, and explain what communication is necessary in order for this process to verify or adjust the partial word count it performs.
Answer: *One way to avoid the problem requires every process except the first process to request the last character from the preceding process. If this "zeroth" character is blank or nonblank, then if the first character is nonblank, the process can correctly decide whether it begins or continues a word.*

d Once each process has done its work, the partial counts need to be combined into a single result. Give a statement in C, C++, or Fortran, including all the arguments, that implements this function call. Indicate which of the arguments to the function represent the actual variables containing the partial and final count information.
Answer: *It is natural to use MPI_Reduce() to gather the individual word counts onto process 0 and sum them. The word counts are single integer scalars, and should all be send to process 0, to be summed into a single scalar with a new name. The call might be*

```
MPI_Reduce ( &my_count, &total_count, 1, MPI_INT,
  MPI_SUM, 0, MPI_COMM_WORLD );   (C)

MPI::COMM_WORLD.Reduce ( &my_count, &total_count, 1,
  MPI::INT, MPI::SUM, 0 );   (C++)

call MPI_Reduce ( my_count, total_count, 1, MPI_INTEGER,
  MPI_SUM, 0, MPI_COMM_WORLD, error )  (Fortran)
```

The variable **my_count** *is the count on a single process, while* **total_count** *will hold the final result.*

2. A character array **t** contains the text of a book, and consists of exactly 10,000,000 characters. The only chararacters are lowercase alphabetic characters ('a' through 'z') and blanks. We wish to construct a histogram, that is, an array **h** of 27 entries, counting the number of times each character occurs in **t**. If you are using C or C++, then h[0] will contain the number of times the letter 'a' occurs in **t**. Moreover, the sum of the entries in **h** will be 10,000,000.

a Write out, in C, C++, or Fortran, a short series of statements that will convert the characters 'a' through 'z' and blank to distinct integer indices. Do something more intelligent than writing 27 **if** statements!
Answer:itblah blah

b Describe an algorithm that will create the histogram, using the indexing you developed in the previous question;
Answer:itblah blah

c Now suppose that we wish to use OpenMP to create the histogram. Write out, in legal C, C++, or Fortran, the loop that will compute the entries of **h**. Include the OpenMP directives that are associated with the loop. You do **not** need to write or display an entire program!
Answer:itblah blah

3. Subdivide the unit square into 4 equal squares of side $\frac{1}{2}$. Consider arrays $x()$ and $y()$ which contain the horizontal and vertical coordinates of n=10,000 points. We wish to count the number of points that lie within each of the 4 squares. That is, we want an array $c()$ of length 4, so that the first entry counts the number of points that fall within the first square, and so on. We might expect that the sum of the entries in c is exactly 10,000, but it might actually be less (or possibly even slightly more!).

a Write a pseudocode program that determines and prints c. Your pseudocode should be written in such a way that an intelligent programmer could use it as a blueprint.
Answer:itblah blah

b Write a partial program, in C, C++, or Fortran, that carries out the same computation as in part a), but does so in parallel, using OpenMP with 4 threads. Include statements that would allow your program to report the execution time for the calculation. You do not have to declare your variables, or worry about the details of print statements or where the x and y arrays come from. The important features of your answer will be the sensible use and ordering of loops, and the correct use and placement of OpenMP directives.
Answer:itblah blah

c Now suppose that you have subdivided the unit square into 100x100 = 10,000 equal subsquares, and you have the coordinates of 7 points. Suppose we wish to create an array $d()$ of length 7, so that each entry of

$d()$ lists the index of the subsquare containing the corresponding point. Write a partial program that carries out this task, using OpenMP with 100 threads.
Answer:itblah blah