Overview

This is a short homework containing problems on recursion and algorithm analysis. It is worth a total of 35 points toward your homework grade, and is due at the start of lab on Wednesday, April 24, 2002. Late homework WILL NOT be accepted! Please DO NOT submit your solutions electronically. Instead, turn in a hand-written or printed version of them.

Questions

1. **(10 points)** Rewrite the recursive function to calculate the sum of the values in a vector using iterators. The prototype is

   ```cpp
   template <class T, class Iter>
   T recursive_sum( Iter begin, Iter end, const T& init );
   ```

   The third argument gives the initial value in the sum. Here’s an example of the function call using code from the lab:

   ```cpp
   vector<int> v;
   v.push_back( 14 ); v.push_back( 15 ); v.push_back( 17 );
   v.push_back( 16 ); v.push_back( 33 ); v.push_back( 27 );
   v.push_back( 12 ); v.push_back( 2 ); v.push_back( 25 );

   int sum = recursive_sum( v.begin(), v.end(), 0.0 );
   ```

   Note that using iterators you usually do not need a second, driver function to initiate the recursion like we’ve needed in many other examples. Also, your function should work for list iterators as well as vector iterators.

2. **(10 points)** Write a recursive function that outputs the digits of an integer in reverse order, one digit per line. Thus, for example, if the number is 12543, the output should be

   3
   4
   5
The function prototype is

```c
void output_reverse( int n );
```

3. **(15 points)** The vector class has a `push_back` member function, but not a `push_front` member function. This question explores why. We will use the `Vec<T>` class that we wrote in class and in lab.

   (a) Write a `push_front` member function for `Vec<T>`. You don’t need to provide the details of reallocation when `data_end == array_bound`, just indicate where in the code it will occur.

   (b) Ignoring the cost of reallocation (doubling the size of the allocated array when `data_end == array_bound`), what is the cost of `push_front`. Briefly justify your answer.

   (c) Ignoring the cost of reallocation (doubling the size of the allocated array when `data_end == array_bound`), what is the cost of `push_back`. Briefly justify your answer.

The difference between the two costs is the reason why vectors do not have a `push_front` function.

Aside: one tough part of the analysis that is ignored here is the cost of the reallocation. Each time reallocation occurs, the cost is \( O(n) \), where \( n \) is the current size of the vector. On the other hand, it doesn’t happen very often. More sophisticated analysis than we can do here shows that the total cost of reallocation for \( m \) consecutive `push_back` operations is \( O(m) \).