Big-Oh Written

Due: October 1, 2001 in class

This assignment is worth 15 points. You can and should discuss this assignment with your classmates. You should write up your own assignment though and mention who you consulted.

1. (2 pts) Suppose $T_1(n) \in O(f(n))$ and $T_2(n) \in O(f(n))$. Answer whether the following are true or false and give justification. A justification for being true is a simple mathematical proof, while you can prove something to be false by giving a specific counterexample. You should use the definition of big-Oh in both cases.

   (a) $T_1(n) + T_2(n) \in O(f(n))$

   (b) $T_1(n) - T_2(n) \in O(f(n))$

   (c) $T_1(n)/T_2(n) \in O(1)$

   (d) $T_1(n) \in O(T_2(n))$
2. (3 pts) Show each of the following false by exhibiting a counterexample. Assume that $f$ and $g$ are any real-valued functions.

(a) If $f(x) \in O(x^3)$ and $g(x) \in O(x)$ then $f(x)/g(x) \in O(x^2)$.

(b) If $f(100) = 1000$ and $f(1000) = 1000000$ then $f$ cannot be $O(1)$.

(c) (extra credit) If $f_1(x), f_2(x), \ldots$ are a bunch of functions that are all in $\Omega(1)$, then the sum

$$F(x) = \sum_{1 \leq i \leq N} f_i(x) \in \Omega(N).$$

If $f(x) \in \Omega(x)$ and $g(x) \in \Omega(x)$ then $f(x) + g(x) \in \Omega(x)$.

3. (3 pts.) Demonstrate the following (that is, demonstrate appropriate constants to plug into the definitions of $O(\cdot)$, $\Omega(\cdot)$, or $\Theta(\cdot)$).

(a) $n! \in \Omega(2^n)$

(b) $n^3 \in \Theta(n^3 + 2n^2)$
(c) (extra credit) \( \sum_{1 \leq i \leq n} 1/i \in O(\log n) \)

\((\log n)^K \in O(n)\) for any \(K\).

4. (2 points) An algorithm takes 0.5ms for input size 100. Assuming no overhead for this initial case, how large a problem can be solved in 1 min if the running time is the following:

(a) linear

(b) \(O(n \log n)\)

(c) quadratic

(d) cubic

5. (3 points) By doubling the size of an array used to store a vector, we pay constant amortized time for each push\_back operation. Suppose that allocating a vector containing \(M\) elements takes \(M/2 + 10\) time units, copying \(M\) elements from one vector to another takes \(M\) time units, and a push takes 1 time unit plus the amount of time (if any) required to increase the size of the vector.

(a) If the size of the vector increases by 100 (that is, 100 more elements, not 100 times as many), how long will \(N\) push\_back operations take?
(b) If the size of the vector doubles each time the stack fills up, how long with \( N \) \textit{push}\_back \ operations take?

(c) If the size of the vector increases by a factor of 1.5 each time, how long will \( N \) \textit{push}\_backs \ take?

6. (2 pts) Assuming we have the following definition of Node

```cpp
struct Node {
    string info;
    Node * next;
    Node(const string & s, Node * follow) // constructor
        : info(s), next(follow) {}
};
```

Fill in the following, using constant extra space.

```cpp
/* True if and only if L is circular. */
bool isCircular (Node *list) // precondition:
// postcondition: list is not changed, returns true if the list is
// circular. Amount of space allocated is constant, i.e. not
// not proportional to the size of the list
{
```