CS109A Quiz

Monday, Oct. 29, 10–10:50 am

Directions

The exam is *open book*; any written materials may be used. Answer all 5 questions on the exam paper itself. The total number of points is 100. You may make additional comments at the bottoms of pages or at the end if you feel the need to clarify or justify your answers. Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

Problem 1: (20 points) The sum of the first n odd integers, $1+3+5+\cdots+(2n-1)$, is exactly n^2. Please prove this result by induction on n.

a) What is a suitable basis case? _______________.

b) Prove the basis case.

c) State the inductive step.

d) Prove the inductive step.

Problem 2: (20 points) Write an ML function `foo` that takes a list of integers and produces the sum of the even-valued elements minus the sum of the odd-valued elements. For example, `foo [10,5,12,8,7]` should produce the value $10+12+8-5-7=18$. It is possible to write this as a single function, and you should strive to do so. However, you may assume the existence of an ML function `ODD` that acts like the Pascal function `ODD`.

Problem 3: (20 points) The class of *wingding strings* is defined recursively as follows.

BASIS: The string 0 is a wingding string.

INDUCTION: If x is a wingding string, so are $x1$ and $x00$.

Less formally, the string of length 1 consisting of a single 0 is a wingding string. If you take a wingding string and append a single 1, you get another wingding string, and if you take a wingding string and append two 0's, you get another wingding string.

a) List all the wingding strings of length four.

b) Write a recurrence for $W(n)$, the number of wingding strings of length n. You need not solve your recurrence, but you must not forget the (one or more) basis cases. *Hint*: If $n > 1$, you must form a wingding string of length n by applying the inductive rule. In what ways can you form wingding strings of length n, and from what length strings are they formed?

Problem 4: (20 points) Consider the recurrence

BASIS: $T(1) = 2$.

INDUCTION: $T(n) = 2T(n-1) - 1$ for integers $n > 1$.

Note that since there are no symbolic constants, this is not a "big-oh" problem; we can solve for $T(n)$ exactly.

a) To begin, what are the initial values of $T(n)$?

$T(1) =$______; $T(2) =$______; $T(3) =$______; $T(4) =$______; $T(5) =$______.

b) Expand the inductive rule so $T(n)$ is expressed in terms of $T(n-2)$.

---

c) Express $T(n)$ in terms of $T(n-3)$.

---

d) What is the general pattern? That is, express $T(n)$ in terms of $T(n-i)$.

---

e) For what value of i can we eliminate $T(n-i)$ from the expression? $i =$________

f) Use your answer to (e) to express $T(n)$ as a function of n alone (i.e., without any terms involving the function T).

$T(n) =$________________.

Problem 5: (20 points) Consider the following fragment of Pascal code. Note that `ODD` is a built-in Pascal function, not a procedure call, and can be assumed to take $O(1)$ time.

```
(1) sum := 0;
(2) for i := 1 to n do
(3)     if ODD(i) then
(4)         for j := 1 to i do
(5)             sum := sum + A[j]
(6)     else
(7)         for j := i to n do
(8)             sum := sum - A[j]
```

Draw a tree (the model is Fig. 3.13 of the notes) showing how the lines are grouped into progressively larger statements. For each statement, indicate the lines it covers and *give a smooth and tight big-oh bound on its running time* (as a function of the value of n).