Sample Questions: Code Run Time Analysis

**August 2015 Computer Science A Question 2 (Iterative Code Segment)**

Consider the following segment of code, assuming that n has been previously declared and initialized to some positive value:

```c
int i, j, k;
for (i = 1; i <= n; i++) {
    for (k = 1; k <= i; k++) {
        j = k;
        while (j > 0) {
            j--;
        }
    }
}
```

(a) (3 pts) Write a summation (3 nested sums) equal to the number of times the statement \( j--; \) executes, in terms of \( n \).

\[
\sum_{i=1}^{n} \left( \sum_{k=1}^{i} \sum_{j=1}^{k} 1 \right)
\]

(b) (7 pts) Determine a closed form solution for the summation above in terms of \( n \).

\[
\begin{align*}
&\sum_{i=1}^{n} \left( \sum_{k=1}^{i} k \right) \\
= &\sum_{i=1}^{n} \left( \sum_{k=1}^{i} \frac{i(i+1)}{2} \right) \\
= &\frac{1}{2} \sum_{i=1}^{n} i^2 + \frac{1}{2} \sum_{i=1}^{n} i \\
= &\frac{1}{2} \cdot \frac{n(n+1)(2n+1)}{3} + \frac{n(n+1)}{4} \\
= &\frac{n(n+1)(4n+5)}{12}
\end{align*}
\]
Dec 14 CSA Q2a
Write a summation, but do NOT solve it, that represents the value of the variable `sum` at the end of the following code segment, in terms of the variable `n`, entered by the user. (Note: your answer should have two summation signs in it and appropriate parentheses that clearly dictate the meaning of the expression you’ve written.)

```c
int i, j, n, sum = 0;
printf("Please enter a positive integer.\n");
scanf("%d", &n);

for (i=n; i<2*n; i++) {
    sum += 1;
    for (j=1; j<=i; j++)
        sum += (j*j);
}
```

$$
\sum_{i=1}^{2n-1} (\sum_{j=1}^{i} j^2)
$$

August 14 CSA Question 2b
Determine the run time of the code segment shown below, in terms of `n`. Provide your answer as a Big-Theta bound.

```c
int n;
scanf("%d", &n);
int i, step = 1, total = 1;

for (i=0; i<n*n; i+= step) {
    total++;
    step += 2;
}
```

<table>
<thead>
<tr>
<th>n</th>
<th>steps</th>
<th>i becomes $n^2-1$, at which point the loop will stop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>15 24</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>9</td>
</tr>
</tbody>
</table>

$$
\sum_{i=1}^{n} (2i-1)
$$

$$
= \left( \sum_{i=1}^{\frac{n(n+1)}{2}} (2i-1) \right) - 1
= 2 \cdot \frac{n(n+1)}{2} - n - 1
= n^2 + n - n - 1
= n^2 - 1
$$
\[
\sum_{i=0}^{2n-1} (1 + \sum_{j=0}^{i-1} 1) = \sum_{i=n}^{2n-1} 1 + \sum_{i=n}^{2n-1} i
\]

\[
= (2n-1-n+1) + \sum_{i=1}^{2n-1} i - \sum_{i=1}^{n-1} i
\]

\[
= n + \frac{(2n-1)2n}{2} - \frac{(n-1)n}{2}
\]

\[
= n + \frac{n}{2} (2 + 2n - (n-1))
\]

\[
= \frac{n}{2} (3n + 1) = \Theta(n^2)
\]
DEC 2013 CSA Q2
(a) (3 pts) Write a summation that represents the number of times the statement `p++` is executed in the following function:

```c
int foo(int n)
{
    int i, j, p = 0;

    for (i = 1; i < n; i++)
        for (j = i; j <= i + 10; j++)
            p++;

    return p;
}
```

(b) (5 pts) Determine a simplified, closed-form solution for your summation from part (a), in terms of `n`. **You MUST show your work.**

Aug 12 CSB Q1
(a) (4 pts) Determine, with proof, the run-time of the following function in terms of the formal parameters `a` and `b`:

```c
int f(int a, int b) {
    int i,j, sum = 0;

    for (i=0; i<a; i++) {
        j = b;
        while (j > 0) {
            j = j/2;
            sum++;
        }
    }
    return sum;
}
```

CS A May 14 Q2a
Write a recurrence relation that represents the runtime of the following function, then solve it (i.e., derive its closed form) using iterative substitution:

```c
int foo(int n)
{
    if (n == 0 || n == 1)
        return 18;

    else
        return foo(n-2) + foo(n-2);
}
```
August 2015 Computer Science B Question 1 (Recursive Code Segment)
Consider the recursive function `diminish` shown below:

```csharp
double diminish(int m, int n)
{
    if (n == 0)
        return m;
    return 1.0/2*diminish(m, n-1);
}
```

(a) (3 pts) Let \( T(n) \) represent the run time of the function `diminish`. Write a recurrence relation that \( T(n) \) satisfies.

\[
T(n) = \begin{cases} 
T(n-1) + O(1) & \text{for } n > 0 \\
\text{perm}(n, k) & \text{for } n = 0, i = 0, \ldots, n \\
\text{perm}(1, n, k, i) & \text{if } i > 0 \\
\end{cases}
\]

(b) (6 pts) Using the iteration method, determine a closed-form solution (Big-Oh bound) for \( T(n) \).

\[
T(n) = \begin{cases} 
T(n-1) + 1 & \text{for } n > 0 \\
T(n-2) + 1 & \text{for } n = 1 \\
\text{perm}(n, k) & \text{for } n = 0 \\
\end{cases}
\]

After \( k \) iterations we have

\[
= T(n-k) + k
\]

Let \( k = n-1 \),

\[
= T(n-(n-1)) + (n-1)
= T(1) + (n-1)
= 1 + (n-1)
= n
= O(n)
\]
Recurrence Relations to Solve

1) \( T(n) = 2T \left( \frac{n}{2} \right) + 1, T(1) = 1 \)

2) \( T(n) = T(n - 1) + n, T(1) = 1 \)

3) \( T(n) = T \left( \frac{n}{2} \right) + n, T(1) = 1 \)

4) \( T(n) = 2T \left( \frac{n}{2} \right) + n, T(1) = 1 \)

Solution to #1 using iteration technique

Original equation: \( T(n) = 2T \left( \frac{n}{2} \right) + 1 \)

Plugging in for \( \frac{n}{2} \), we get \( T \left( \frac{n}{2} \right) = 2T \left( \frac{n}{4} \right) + 1 = 2T \left( \frac{n}{4} \right) + 1 \)

Similarly, we find:

\[
T(n) = 2T \left( \frac{n}{2} \right) + 1 \\
= 2 \left( 2T \left( \frac{n}{4} \right) + 1 \right) + 1 \\
= 4T \left( \frac{n}{4} \right) + 2 + 1 \\
= 4T \left( \frac{n}{4} \right) + 3
\]

Repeat, plugging in \( T \left( \frac{n}{4} \right) \):

\[
= 4 \left( 2T \left( \frac{n}{8} \right) + 1 \right) + 3 \\
= 8T \left( \frac{n}{8} \right) + 4 + 3 \\
= 8T \left( \frac{n}{8} \right) + 7
\]

In general, after \( k \) steps, we get:

\[
T(n) = 2^k T \left( \frac{n}{2^k} \right) + (2^k - 1)
\]

If we let \( 2^k = n \) (so that \( k = \log_2 n \)), we get

\[
T(n) = nT \left( \frac{n}{n} \right) + (n - 1) = n(1) + (n - 1) = 2n - 1 = O(n)
\]

Yielding the Big-Oh bound of the recurrence relation.