CPE 360: Homework 1

Mukund Iyengar
miyengar@stevens.edu

Only **hardcopy submissions** are accepted for this homework since much of it is handwritten. Co-operation is allowed, but blatant copying is not acceptable.

1. Prove, or disprove, each of the following:

   (a) $2n + 9 = O(1)$

   (b) $2^{n+3} = O(2^n)$

   (c) $3n^2 + 4n + 5 = O(n^2)$

   (d) $n + n \times \log n = O(n)$

   (e) $2n + 3n^2 + 4n^3 + n \times \log n = O(n\log(n))$

2. Give an analysis of the running time (Big-O) for the following code snippets:

   (a)
   ```
   sum = 0;
   for(i = 0; i < n; i++)
      sum++;
   ```

   (b)
   ```
   sum = 0;
   for(i = 0; i < n; i++)
     for(j = 0; j < n; j++)
       sum++;
   ```

   (c)
   ```
   sum = 0;
   for(i = 0; i < n; i++)
     for(j = 0; j < n*n; j++)
       sum++;
   ```

   (d)
   ```
   sum = 0;
   for(i = 0; i < n; i++)
     for(j = 0; j < i; j++)
       sum++;
   ```

   (e)
   ```
   sum = 0;
   for(i = 0; i < n; i++)
     for(j = 0; j < i*i; j++)
        for(k = 0; k < j; k++)
           sum++;
   ```

3. Consider the following code fragment that uses recursion to implement a function called `processThis()`

```
int processThis(int n)
{
  if(n == 0)
    return 1;
  else
   return processThis(n/2 + 1);
}
```

Assume we call this function from a program with $n = 5$, i.e., we call `processThis(5)`. What do you think is wrong with this code, and why?

4. Analyze the following program which recursively solves the "Towers of Hanoi" problem for time complexity. Prove that the number of moves required is of the form $O(2^n)$.

```
void towersHanoi(int N, char S, char A, char D)
{
  if(N == 1)
      print "Move disc 1 from S --> D";
  else
  {
      towersHanoi(N-1, S, A, D);
      print "Move disc N from S --> D";
      towersHanoi(N-1, A, D, S);
  }
}
```

where `N` denotes the number of discs, and the three pegs are marked source (`S`), destination (`D`) and auxiliary (`A`), respectively.