Example Class 1 (Week 4 – Week 5)

Each group will have TWO students to present for ONE of the following questions.

Perform complexity analysis for the algorithm by (1) find the recurrent equation; (2) solve the recurrent equation and (3) conclude on the complexity class for the algorithm by a proof. The presentation should include how the recurrent equation is derived, how it is solved, and prove the asymtotic complexity.

For this session, you should be able to prepare for your presentation completely within the first 30 minutes of your class.

1. Analyze the number of disk moves in terms of n (to be done by group 1)

```
void TowersOfHanoi(int n, int x, int y, int z)   // n>= 0
{ // Move n disks from tower x to tower y.
  // Use tower z for intermediate storage.
   if (n > 0) {
      TowersOfHanoi(n-1, x, z, y);
      cout << "Move  disk from  " << x
           << " to  " << y << endl;
      TowersOfHanoi(n-1, z, y, x);}
}
```

2. Analyze the number of multiplications in terms of n. (to be done by group 2)

```
int sumFactorials(n)
{ // get the sum from factorial(1) to factorial(n)
  // n is a positive integer

        if (n == 1) return 1;
        else {
                f = 1;
                for (j = 2 to n)  f = f* j;
                return f + sumFactorials(n-1);
        }
}
```

3. Analyze the number of additions in terms of n. Assume that $n = 2^k$. (to be done by group 3)

```
int binaryDigits(int n)
{ // count the number of binary digits in n's binary representation.
  // n is a positive decimal integer

        if (n == 1) return 1;
        else return binaryDigits(⌊n/2⌋) + 1;
}
```

4. Analyze the number of comparisons between array elements (between min1 and min2 throughout the recursive calls) in terms of n, the size of the array A. Assume that $n = 2^k$. (to be done by group 4)

```
int minimum(A[l..r])
{// compute the minimum value in the array A starting from position l till position r
 // n = r-l+1 and l <= r

        if (l == r)  return A[ l ]
        else {
                min1 = minimum(A[ l..⌊(l+r)/2⌋ ]);
                min2 = minimum(A[ ⌊(l+r)/2⌋+1..r ]);
                if (min1 < min2) return min1
                else return min2;
        }
}
```

5. For the function minimum() in Question 4, analyze the number of divisions in terms of n. Assume that $n = 2^k$. (to be done by group 5)

6. Analyze the total number of additions and subtractions in terms of n. (to be done by group 6)

```
int Q(int n)
{ // n is a positive integer

        if (n == 1) return 1;
        else return Q(n-1) + 2*n – 1;
}
```

Students on medical certificate or leave of absence: please submit a hand-written solution to a question which is not the one done by your group.