Lab 5 – Cost Prediction & Measurement

Cost Prediction

In this section, we will use basic COCOMO to predict the cost of a software development project. Given the following table:

Task	Predecessors	Optimistic	Normal	Pessimistic	Estimated LOC
A	-	1	2	3	202
B	A	1	4	5	435
C	B	6	7	8	788
D	B	3	4	4	420
E	B	2	6	7	600
F	E	1	3	4	340
G	F	9	9	9	1200
H	G	1	2	3	197

Task 1

Given the associated COCOMO coefficients as follows:

Development Context	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Suppose the development context is chosen as organic, try to estimate the effort E, the development time D and the required number of people P using COCOMO for each of the tasks (A-H).

To compute **E**, **D**, and **P**, you need to use the following equations:

$$E = a * S^b$$

$$D = c * E^d$$

$$P = E/D$$

Note, the unit of S is not LOC, but $KLOC$, which can be computed as:

$$KLOC = \frac{LOC}{1000}$$

Since we use organic development context, the relevant COCOMO coefficients are as follows:

$$a = 2.4, b = 1.05, c = 2.5, d = 0.38$$

For instance, for task A, first compute $KLOC = \frac{202}{1000} = 0.202$, and then conduct the following computations:

$$E = 2.4 * 0.202^{1.05} = 0.45$$

$$D = 2.5 * 0.45^{0.38} = 1.85$$

$$P = \frac{0.45}{1.85} = 0.24 = 1\ person$$

because a person is an integer, and we need to have an upper bound to it even though P can have decimal (that's why we will need to round up the number)

Similarly, you could compute the **E**, **D**, and **P** for all the remaining tasks. If you done it correctly, you will get the following results:

Task	E	D	P
A	0.45	1.85	1
B	1.0	2.5	1
C	1.87	3.17	1
D	0.97	2.47	1
E	1.4	2.84	1
F	0.77	2.26	1
G	2.91	3.75	1
H	0.44	1.83	1

Measurement

In this section, you are going to measure the size or the complexity of a software development project. Given the following java code:

```java
/*Given a binary array, find the maximum number of
consecutive 1s in this array.*/
public int findMaxConsecutiveOnes(int[] nums) {
    int max = 0;
    boolean flag = false;
    int count = 0;
    for (int i = 0; i < nums.length; i++) {
      if (nums[i] == 1) {
        if (!flag) {
          flag = true;
        }
        count++;
        max = Math.max(max, count);
      } else {
        count = 0;
        flag = false;
      }
    }
    return max;
  }
```

You are required to accomplish the following tasks:

Task 2

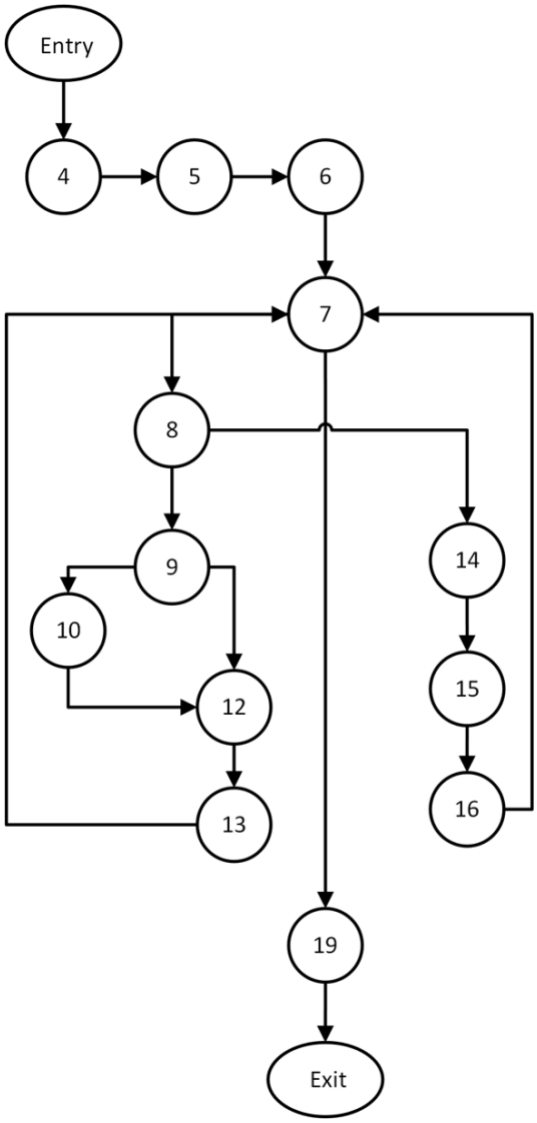
Compute the basic lines of codes (LOC) and the percentage of the LOC that are comments (COM).

To compute LOC, you need to count how many lines in the above code snippet. For this simple code snippet, $LOC = 20$ (*this includes the code comments*). Furthermore, to compute COM, you need to count how many lines of comments existed in the above code snippet. In this example, there are 2 lines of comments. Thus, $COM = \frac{2}{LOC} = \frac{2}{20} = 0.1$.

Task 3

Compute the corresponding cyclomatic complexity (CC).

To compute CC for the above code snippet, you need to draw the corresponding control flow graph, which is demonstrated as follows:



Within the above control flow graph, there are $N = 15$ nodes (*including Entry and Exit*) and $E = 17$ edges, and you are dealing with $P = 1$ procedure. Therefore, the resulting CC can be computed as follows:

$$CC = E - N + 2P = 17 - 15 + 2 * 1 = 4$$