Assignment: Find $T(n)$ for code fragment 3 given in Figure 1. Put the results of your analysis in the submission to your instructor. Your analysis should be patterned after the analysis in Figure 6. Transform the code fragment into while-loops. Account for the cost of each line. Sum the total and express it in closed form.

Write function *af3* that returns $T(n)$ according to your analysis. Functions *af3* computes $T(n)$ analytically. Model your design of functions *af3*, after example analytical function *af0* shown in Figure 4.

Write functions *ef3* that returns $T(n)$ by counting the number of operations that are executed in the code fragment. Function *ef3* computes $T(n)$ empirically. Model your design of functions *ef3* after example empirical function *ef0* shown in Figure 5.

Deliverables:

1. A three-column analysis of the code fragment. Put the analysis in appropriate sections of the template (http://cs2.uco.edu/~trt/cs2123/p03ProjectTemplate.docx) created for them below the author identification in the Microsoft Word document you submit to your instructor.
2. A copy of all the source files in your project in the Microsoft Word document you submit to your instructor.
3. A copy of all the source files in the root directory of your student account.

Prohibition: Use of the C++ Standard Template Library is prohibited in the implementation of this project.

Program Files: Project **3** consists of the single file **p03.cpp.** You must name your source file **p03.cpp**. Failure to name your file **p03.cpp** will result in a score of ***zero*** (***0***) for this project

Project files must be stored in the **root directory of your student account**. Failure to store project files in the root directory of your student account will result in a score of ***zero*** (***0***) for this project.

File	Description
p03.cpp	File **p03.cpp** contains functions that process command line arguments. File p03 also contains time complexity functions.

Command Line: Project **3** can be invoked with zero, one, or two program parameters. The first program parameter is the input file name. The second parameter is the output file name. Sample command lines together with corresponding actions by program **p03** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

$ **p03**
Enter the first input file name: **i03.dat**
Enter the output file name: **o03.dat**

$ **p03 i03.dat**
Enter the output file name: **o03.dat**

$ **p03 i03.dat o03.dat**

Input files: Files **i03.dat** contains values of n for code fragments 3. Program **p03** reads the input file and produces output for that file. The input file consists of a sequence of integers. Integers are separated by white space. White space is one or more characters from the set that includes a blank character, a newline character, or a tab character.

Example Input File 0 1 2 3 4 5 6 7 8 9 10

Output files: Print your results in the output file.

1. Print a line identifying the code fragment. For example, this code fragment is identified as
   `Code Fragment 3`
2. Put your output in three columns. Label the first column **n**, the second column **Analytical** and the third column **Empirical.** Produce values of $T(n)$ for each code fragment. One line is printed for each value of n read. Write the value of n in the column labeled **n**. Write the value of $T(n)$ computed by the designated analytical function, *af3*, in the column labeled **Analytical**. Write the value of $T(n)$ computed by the selected empirical function, *ef3*, in the column titled **Empirical**. Right justify and align data produced in rows with their corresponding columns.

```
int sum=1;
for (int a=0;a<n;a++) sum=sum*2;
while (sum>0) sum--;
```

Figure 1. Code Fragment 3

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

Figure 2. Example Code Fragment 0

```
Code Fragment 0
    N      Analytical       Empirical
    0               3               3
    1               6               6
    2               9               9
    3              12              12
    4              15              15
    5              18              18
    6              21              21
    7              24              24
    8              27              27
    9              30              30
   10              33              33
```

Figure 3. File o03.dat – Example Output (for Code Fragment 0)

```
int af0(int n){return 3*n+3;}
```

Figure 4. Analytical Timing Function *af0*.

```
int ef0(int n)
{   int t=0;
    int sum=0;      t++;
    int i=0;        t++;
    while (i<n) {   t++;
        sum++;      t++;
        i++;        t++;
    }               t++;
    return t;
}
```

Figure 5.Empirical Timing Function *ef0*.

Line	Code	Cost
1	*sum*=0;	1
2	*i*=0;	1
3	while (*i*<*n*) {	$\sum_{i=0}^{n-1} 1 = n$
4	*sum*++;	$\sum_{i=0}^{n-1} 1 = n$
5	*i*++;	$\sum_{i=0}^{n-1} 1 = n$
6	}	1
	Total	$\boldsymbol{T(n) = 3n + 3}$

Figure 6. Analysis of Code Fragment 0