**Project:**

Project **2** contains time complexity analysis for the six code fragments given in Figures 1 to 6. Find $T(n)$, $f(n)$, $c$, $n_0$ and $O(f(n))$ for each of the code fragments given in Figures 1 to 6. Put the results of your analysis in a document immediately following your cover sheet. Your analysis should be patterned after the analysis in Figure 11. Transform the code fragment into *while-statements*. Account for the cost of each line. Sum the total and express it in closed form.

Write functions *af01, af02, ..., af06* that return *T(n)* according to your analysis. Functions *af01, af02, ..., af06* compute *T(n)* analytically.

Write functions *ef01, ef02, ..., ef06* that return *T(n)* by counting the number of times each statement in the code fragment executes. Functions *ef01, ef02, ..., ef06* compute *T(n)* empirically.

**Deliverables:**

| **Item** | **Description** |
|---|---|
| **Time complexity analyses** | Submit time complexity analyses for each of the six code fragments given in figures 1 – 6. Find $T(n)$, $f(n)$, $c$, $n_0$ and $O(f(n))$ for each code fragment. Your analysis should be patterned after the analysis in Figure 11. Put your analyses in a Microsoft ® Word document. Employ the mathematical formatter to express mathematical relationships you discover in your analyses. |
| **Programs** | Submit program **p02** and all constituent files as described below. |

**Program Files:**

| **File** | **Description** |
|---|---|
| **p02.cpp** | File **p02.cpp** contains functions that process command line arguments and exercise analytical and empirical functions. |
| **Sum02.h** | File **Sum02.h** contains function prototypes that find the sum of finite sequences $\sum_{i=1}^{n} 1, \sum_{i=1}^{n} i, \sum_{i=1}^{n} i^2, \sum_{i=1}^{n} i^3$, and $\sum_{i=1}^{n} i^4$ |
| **Sum02.cpp** | File **Sum02.cpp** implements functions whose prototypes are contained in file **Sum02.h**. |
| ***02.h** | Create as many classes as you feel are necessary. The file names of all header files for this project must have the suffix **02.h**. |
| ***02.cpp** | Create as many classes as you feel are necessary. The file names of all the class implementations must have the suffix **02.cpp**. |
| **p02make** | File **p02make** contains instructions for program **p02**. Instructions are written for the UNIX utility ***make***. Program **p02** is contained in file **p02**. |

**Command Line:** Project **2** can be invoked with up to seven program parameters. The first six program parameters are names of input files. The seventh program parameter is the name of the output file. Sample command lines together with corresponding actions by program **p02** are shown below. Boldfaced type indicates data entered at the keyboard by the user.

$ **p02**
Enter input file number 1: **i021.dat**
Enter input file number 2: **i022.dat**
Enter input file number 3: **i023.dat**
Enter input file number 4: **i024.dat**
Enter input file number 5: **i025.dat**
Enter input file number 6: **i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat**
Enter input file number 2: **i022.dat**
Enter input file number 3: **i023.dat**
Enter input file number 4: **i024.dat**
Enter input file number 5: **i025.dat**
Enter input file number 6: **i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat i023.dat**
Enter input file number 4: **i024.dat**
Enter input file number 5: **i025.dat**
Enter input file number 6: **i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat i023.dat i024.dat**
Enter input file number 5: **i025.dat**
Enter input file number 6: **i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat i023.dat i024.dat i025.dat**
Enter input file number 6: **i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat i023.dat i024.dat i025.dat i026.dat**
Enter the output file name: **o02.dat**

$ **p01 i021.dat i022.dat i023.dat i024.dat i025.dat i026.dat o02.dat**

**Input Files:** Files **i021.dat, i022.dat, i023.dat, i024.dat, i025.dat**, and **i026.dat** contain values of *n* for code fragments (1), (2), (3), (4), (5), and (6) respectively. Program **p02** reads an input file and produces output for that file. Program **p02** reads input files in succession and produces output for each input file as the file is read.

**Output File:** Send your output to both the output file and the display (*stdout*). 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, *af0x*, in the column labeled **Analytical**. Write the value of *T(n)* computed by the selected empirical function, *ef0x*, in the column titled **Empirical**. Produce one table for each code fragment. Identify the tables by the code fragment analyzed. Figure 7 contains the output for code fragment 0.

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

**Figure 1**. Code Fragment 1

```
for (i=0;i<n;i++) {
    m=n;
    while (m>1) m=m/2;
}
```

**Figure 2**. Code Fragment 2

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

**Figure 3**. Code Fragment 3

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

**Figure 4**. Code Fragment 4

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

**Figure 5**. Code Fragment 5

```
sum=0;
for(i=1;i<n;i++)
    for(j=1;j<i*i;j++)
        if (j% i==0)
            for(k=0;k<j;k++)
                sum++;
```

**Figure 6**. Code Fragment 6

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

**Figure 7**. Code Fragment 0

```
int f::af00(int n)
{    return 3*n+3;
}
```

**Figure 8.** Analytical Timing Function (*af00*) for Code Fragment 0

```
int f::ef00(int n)
{    int i,sum,t=0;
     sum=0;              t++;
     i=0;                t++;
     while (i<n) {       t++;
          sum++;         t++;
          i++;           t++;
     }                   t++;
     return t;
}
```

**Figure 9**. Empirical Timing Function (ef00) for Code Fragment 0

| n | Analytical | Empirical |
|---|---|---|
| 0 | 3 | 3 |
| 10 | 33 | 33 |
| 20 | 63 | 63 |
| 30 | 93 | 93 |
| 40 | 123 | 123 |
| 50 | 153 | 153 |
| 60 | 183 | 183 |
| 70 | 213 | 213 |
| 80 | 243 | 243 |
| 90 | 273 | 273 |

**Figure 10.** Output for Code Fragment 0

| Line | Code | Cost | Reduced Form |
|---|---|---|---|
| 1 | *sum*=**0;** | 1 | 1 |
| 2 | *i*=**0;** | 1 | 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 | 1 |
| | | $T(n) =$ | $3n+3$ |

Select $f(n) = n$

$$c_{\min} = \lim_{n\to\infty} \frac{T(n)}{f(n)} = \lim_{n\to\infty} \frac{3n+3}{n} = 3$$

$$c = 4$$

Solve $T(n_0) \le cf(n_0)$

$$3n_0 + 3 \le 4n_0 \Rightarrow n \ge 3$$

**Figure 11.** Analysis of Code Fragment 0

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^{n} k^3 = \left[\frac{n(n+1)}{2}\right]^2$$

$$\sum_{k=1}^{n} k^4 = \frac{1}{30} n(n+1)(2n+1)(3n^2+3n-1)$$

**Figure 12.** Finite sums.