Author Identification Block

Author: Mr. Alan Turing
Student ID: *00000000
E-Mail: aturing@uco.edu
Course: CMSC 2613 – Programming II
CRN: 21641, Spring, 2013
Project: p09
Due: April 22, 2013
Account: tt000

Scoring Block			
Component	**Available**	**Earned**	**Explanation**
Compilation			
Submission Instructions	2	2	
Author Identification	1	1	
Fragment 1 Analysis	3	3	
Fragment 2 Analysis	3	3	
Fragment 3 Analysis	3	3	
Command Line	2	2	
Output file	2	2	
Fragment 1 Results	3	3	
Fragment 2 Results	3	3	
Fragment 3 Results	3	3	
Total	**25**	**25**	

Code Fragment 1 Analysis:

1. Find $T(n)$

```
int sum=0;
for (i=0;i<n;i++) {
    int m=n;
    while (m>1) m=m/3;
}
```

Line	Code	Cost
1	**int** *sum*=**0;**	$\mathbf{1}$
2	**int** *i*=**0;**	$\mathbf{1}$
3	**while (***i*<*n***) {**	$a = \sum_{i=0}^{n-1} 1 = n$
4	**int** *m*=*n***;**	a
5	**while (***m*>**1) {**	$b = \sum_{i=0}^{n-1} \sum_{m=1}^{\lfloor \log_3 n \rfloor} 1$
6	*m*=*m*/**3;**	$2b$
7	**}**	a
13	*i*++**;**	a
14	**}**	$\mathbf{1}$
$T(n) = 3b + 4a + 3$		
$T(n) = 3\sum_{i=0}^{n-1} \sum_{m=1}^{\lfloor \log_3 n \rfloor} 1 + 4n + 3$		
$b = \sum_{i=0}^{n-1} \sum_{m=1}^{\lfloor \log_3 n \rfloor} 1 = n\lfloor \log_3 n \rfloor$		
$T(n) = 3n\lfloor \log_3 n \rfloor + 4n + 3$		

2. Find $f(n)$

$f(n) = n\lfloor \log_3 n \rfloor$

3. Find C

$C = C_{min} + \Delta = 3 + 1 = 4$

4. Find n_0

i. $|T(n_0)| \leq C|f(n_0)|$

ii. $3n_0\lfloor\log_3 n_0\rfloor + 4n_0 + 3 \leq 4n_0\lfloor\log_3 n_0\rfloor$

iii. $4n_0 + 3 \leq n_0\lfloor\log_3 n_0\rfloor$

n_0	$4n_0 + 3$	$\leq$	
3	15	No	3
9	39	No	18
27	111	No	81
81	327	No	324
82	331	No	328
83	335	No	332
84	339	No	336
85	343	Yes	343

Conclusion: $n_0 = 85$

Therefore, $T(n)$ is $O(n\lfloor\log_3 n\rfloor)$ because we have found witnesses, positive values, $C = 4$ and $n_0 = 85$ that make $|T(n)| \leq C|f(n)|$ where $f(n) = n\lfloor\log_3 n\rfloor$ and $n \geq n_0$.

Code Fragment 2 Analysis:

Code Fragment 3 Analysis:

File p09.cpp

```
//-----------------------------------------------------------------------
//File p09.cpp processes command line arguments and directs the execution
//of empirical and analytical measures of time complexity for code fragments
//given in programming project p09, CMSC 2613 - Programming II
//-----------------------------------------------------------------------
//File     Description
//p09.cpp  Processes command line arguments and directs execution of empirical
//         and analytical measures of time complexity for code fragments
//         given in programming project p09, CMSC 2613 - Programming II
//F09.h    Defines interfaces to time complexity measures
//F09.cpp  Implements complexity measures.
//-----------------------------------------------------------------------
//Author:  Thomas R. Turner
//E-Mail:  trturner@uco.edu
//Date:    April, 2016
//-----------------------------------------------------------------------
//Copyright April, 2016 by Thomas R. Turner.
//Do not reproduce without permission from Thomas R. Turner
//-----------------------------------------------------------------------
//Standard C and C++ include files
//-----------------------------------------------------------------------
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
using namespace std;
//-----------------------------------------------------------------------
//Application include files
//-----------------------------------------------------------------------
#include "F09.h"
//-----------------------------------------------------------------------
//Function Title prints the title for a code fragment
//-----------------------------------------------------------------------
void Title(int cf,ostream& o)
{   o << endl;
    o << "Code Fragment " << cf;
    o << endl;
    o << setw( 5) << "n";
    o << setw(15) << "empirical";
    o << setw(15) << "analytical";
}
```

```
//---------------------------------------------------------------------------
//Function Print prints n, empirical, and analytical values
//---------------------------------------------------------------------------
void RowPrint(ostream& o,int n,int e,int a)
{   o << endl;
    o << setw( 5) << n;
    o << setw(15) << e;
    o << setw(15) << a;
}
//---------------------------------------------------------------------------
//Function TimeComplexityMgr directs the measurement of empirical and
//analytical time complexity for code fragments 1, 2, and 3.
//---------------------------------------------------------------------------
void TimeComplexityMgr(istream& i1,istream& i2,istream& i3,ostream& o)
{   Title(1,o);
    for (;;) {
        int n;
        i1 >> n;
        if (i1.eof()) break;
        RowPrint(o,n,Empirical01(n),Analytical01(n));
    }
    o << endl;
}
//---------------------------------------------------------------------------
//A FileException is thrown when a file whose name appears on the
//command line cannot be opened.
//---------------------------------------------------------------------------
struct FileException {
    FileException(char* fn)
    {   cout << endl;
        cout << "File " << fn << " could not be opened.";
        cout << endl;
    }
};
```

```
//---------------------------------------------------------------------
//A CommandLineException is thrown when too many arguments appear on
//the command line.
//---------------------------------------------------------------------
struct CommandLineException {
   CommandLineException(int m, int a)
   {    cout << endl;
        cout << "Too many file names on the command line.";
        cout << endl;
        cout << "A maximum of " << m << " file names can appear on the "
             << "command line.";
        cout << endl;
        cout << a << " file names were on the command line.";
        cout << endl;
   }
};
//---------------------------------------------------------------------
//Function main processes command line arguments.
//---------------------------------------------------------------------
int main(int argc, char* argv[])
{
   try {
      char ifn1[255];     //Input file name 1
      char ifn2[255];     //Input file name 2
      char ifn3[255];     //Input file name 3
      char ofn [255];     //Output file name
      switch (argc) {
         case 1:         //Prompt for all file names
            cout << "Enter input file name 1: ";
            cin  >> ifn1;
            cout << "Enter input file name 2: ";
            cin  >> ifn2;
            cout << "Enter input file name 3: ";
            cin  >> ifn3;
            cout << "Enter the output file name. ";
            cin  >> ofn;
         break;
         case 2:         //Copy input file name 1 from argv. Prompt for the remainder of the file names.
            strcpy(ifn1,argv[1]);
            cout << "Enter input file name 2: ";
            cin  >> ifn2;
            cout << "Enter input file name 3: ";
            cin  >> ifn3;
            cout << "Enter the output file name. ";
            cin  >> ofn;
         break;
```

```
        case 3:     //Copy input file names 1 and 2 from argv. Prompt for the remainder of the file
                    // names.
          strcpy(ifn1,argv[1]);
          strcpy(ifn2,argv[2]);
          cout << "Enter input file name 3: ";
          cin  >> ifn3;
          cout << "Enter the output file name. ";
          cin  >> ofn;
        break;
        case 4:     //Copy all input file names from argv. Prompt for the output file name.
          strcpy(ifn1,argv[1]);
          strcpy(ifn2,argv[2]);
          strcpy(ifn3,argv[3]);
          cout << "Enter the output file name. ";
          cin  >> ofn;
        break;
        case 5:     //Copy all file names from argv.
          strcpy(ifn1,argv[1]);
          strcpy(ifn2,argv[2]);
          strcpy(ifn3,argv[3]);
          strcpy(ofn ,argv[4]);
        break;
        default:    //Too many file names on the command line.
          throw CommandLineException(4,argc-1);
        break;
      }
      ifstream i1(ifn1); if (!i1) throw FileException(ifn1);
      ifstream i2(ifn1); if (!i1) throw FileException(ifn1);
      ifstream i3(ifn1); if (!i1) throw FileException(ifn1);
      ofstream o (ofn ); if (!o ) throw FileException(ofn );

      TimeComplexityMgr(i1,i2,i3,o);
      o.close(); //Close the output file
      i3.close();//Close input file 3
      i2.close();//Close input file 2
      i1.close();//Close input file 1
    } catch( ... ) {
      cout << endl;
      cout << "Program terminated!";
      cout << endl;
      cout << "I won't be back!";
      cout << endl;
      exit(EXIT_FAILURE);
    }
    return 0;
}
```

File F09.h

```
#ifndef F09_h
#define F09_h 1
//-----------------------------------------------------------------------
//File F09.h Defines interfaces to time complexity measures
//-----------------------------------------------------------------------
//File     Description
//p09.cpp  Processes command line arguments and directs execution of empirical
//         and analytical measures of time complexity for code fragments
//         given in programming project p09, CMSC 2613 - Programming II
//F09.h    Defines interfaces to time complexity measures
//F09.cpp  Implements complexity measures.
//-----------------------------------------------------------------------
//Author:  Thomas R. Turner
//E-Mail:  trturner@uco.edu
//Date:    April, 2016
//-----------------------------------------------------------------------
//Copyright April, 2016 by Thomas R. Turner.
//Do not reproduce without permission from Thomas R. Turner
//-----------------------------------------------------------------------
//Standard C and C++ include files
//-----------------------------------------------------------------------
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
using namespace std;
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
int Analytical01(int n);
int Empirical01(int n);
int Analytical02(int n);
int Empirical02(int n);
int Analytical03(int n);
int Empirical03(int n);
#endif
```

File F09.cpp

```
//-----------------------------------------------------------------------
//File F09.cpp Implements complexity measures.
//-----------------------------------------------------------------------
//File     Description
//p09.cpp  Processes command line arguments and directs execution of empirical
//         and analytical measures of time complexity for code fragments
//         given in programming project p09, CMSC 2613 - Programming II
//F09.h    Defines interfaces to time complexity measures
//F09.cpp  Implements complexity measures.
//-----------------------------------------------------------------------
//Author:  Thomas R. Turner
//E-Mail:  trturner@uco.edu
//Date:    April, 2016
//-----------------------------------------------------------------------
//Copyright April, 2016 by Thomas R. Turner.
//Do not reproduce without permission from Thomas R. Turner
//-----------------------------------------------------------------------
//Standard C and C++ include files
//-----------------------------------------------------------------------
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <cmath>
using namespace std;
//-----------------------------------------------------------------------
//Application include files
//-----------------------------------------------------------------------
#include "F09.h"
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
double log3(double x){return log(x)/log(3.0);}
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
int Analytical01(int n)
{   if (n>0)
       return 3*n*(int)round(log3((double)n))+4*n+3;
    else
       return 3;
}
```

```
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
int Empirical01(int n)
{   int c=0;
    int sum=0;              c++;
    int i=0;                c++;
    while (i<n) {           c++;
       int m=n;             c++;
       while (m>1) {        c++;
          m=m/3;            c+=2;
       }                    c++;
       i++;                 c++;
    }                       c++;
    return c;
}
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
int Analytical02(int n){return 0;}
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
int Empirical02(int n){return 0;}
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
int Analytical03(int n){return 0;}
//--------------------------------------------------------------------------
//--------------------------------------------------------------------------
int Empirical03(int n){return 0;}
```

File p09make #----------

```
# File p09make create executable file p09 from source files p09.cpp,
# F09.h, and F09.cpp
#-----------------------------------------------------------------
# Author: Thomas R. Turner
# E-Mail: trturner@uco.edu
# Date:   April, 2016
#-----------------------------------------------------------------
# Copyright April, 2016 by Thomas R. Turner.
# Do not reproduce without permission from Thomas R. Turner.
#-----------------------------------------------------------------
# Object files
#-----------------------------------------------------------------
obj     =       p09.o F09.o
#-----------------------------------------------------------------
# Bind object files into executable file p09
#-----------------------------------------------------------------
p09:            ${obj}
            g++ -o p09 ${obj} -lm

#-----------------------------------------------------------------
# Compile file p09.cpp
#-----------------------------------------------------------------
p09.o:          p09.cpp F09.h
            g++ -c -g p09.cpp
#-----------------------------------------------------------------
# Compile file F09.cpp
#-----------------------------------------------------------------
F09.o:          F09.cpp F09.h
            g++ -c -g F09.cpp
```