CSC 840

Matthew Wishoff

4/2/2016

Project 5

Table of Contents

1.0 Abstract

What we plan to do for this project is develop a tool that will help calculate a constructive cost model for software engineering projects. I decided to do my COCOMO project in C++. Some of the challenges that I will face is converting the kilobytes of an executable into lines of code. However when given the KLOC of a program I know I will be able to exactly calculate the estimations in the constructive cost model. The reason why I think it will be challenging is because I've never done it before, and I'm not sure how accurate the formula will be. However something constant with this experiment is the formulas we are using to calculate Effort, Productivity, Time, Staffing, and development cost.

2.0 Introduction

Looking at the Constructive Cost Model this will be a very valuable tool to use in industry practice as a manager. It gives a person the ability to estimate the time needed to do a certain project, as well as how many people are needed, and how much it will cost the company to develop. The tool gives you the options of Organic, Semidetached, or Embedded which are three different environments to develop software in. It allows the user to be flexible in how they wish to apply the tool, so it can be used for almost any situation. Next we also have the drivers which allow the user to specify the criteria in which the software will be developed. You can alter things such as Product attributes, Computer Attributes, Personnel Attributes, and Project Attributes. Being able to change all these things increases the flexibility of the COCO Model to be able to compute for a wide variety of environments and projects.

3.0 Method and software used

For this project I used Dev-C++ to develop the code for the Constructive Cost Model that I wrote in C++. I took programs with varying executable sizes and used the COCOMO program I made to gauge their development time and cost. I tried to pick programs that I used frequently to get a better feel for how much time and effort went into these applications. The popular applications I chose were Hearthstone a popular video game I play frequently, Battlenet which is a platform used to connect gamers to blizzard games, PowerPoint, Excel, Notepad++, Steam which is a platform used to connect gamers to a wide variety of games, Microsoft malicious software removal tool (MSRT), and finally the Java script operating system my team made in CSC 415.

4.0 Graphs and Data

Drivers\Applications	Hearthstone	Steam	Notepad++	BattleNet
Required Software Reliability	Very high	Very high	Normal	Very high
Data base size	High	Very high	Normal	High
Product Complexity	High	Low	Normal	High
Execution time constraint	Normal	High	Normal	Normal
Main storage constraint	Normal	Normal	Normal	Normal
Virtual machine Volatility	Normal	Normal	Normal	Normal
Computer turnaround time	Very high	High	Normal	Very high
Analytical Capabilities	Very high	Very high	Normal	Very high
Applications experience	Very high	Very high	Normal	Very high
Programmer Capability	Very high	Very high	Normal	Very high
Virtual Machine experience	High	Normal	Normal	High
Programming lang. experience	High	High	Normal	High
Using modern Prog. Practices	Very high	Normal	Normal	Very high
Use of software tools	Very high	Normal	Normal	Very high
Required Development Schedule	Low	Normal	Normal	Low

Drivers\Applications	Excel	PowerPoint	MSRT	JavaScript OS
Required Software Reliability	Normal	Very high	Very high	Low
Data base size	Normal	Low	Low	Low
Product Complexity	Normal	Normal	Extra high	Extra high
Execution time constraint	Normal	Normal	Very high	Normal
Main storage constraint	Normal	Normal	High	Normal
Virtual machine Volatility	Normal	Normal	High	Low
Computer turnaround time	Very high	Very high	Very high	Normal
Analytical Capabilities	Very high	Very high	Very high	Very high
Applications experience	Very high	Very high	Very high	Very low
Programmer Capability	Very high	Very high	Very high	Very high
Virtual Machine experience	High	High	High	Low
Programming lang. experience	High	High	High	Very low
Using modern Prog. Practices	Very high	Very high	Very high	Very high
Use of software tools	Very high	Very high	Very high	Very high
Required Development Schedule	Low	Low	Very low	Very high

4.1 COCOMO Input

COCOMO Input	Hearthstone	Steam	Notepad++	Battlenet
Executable size in kb	15,733	3,006	2,015	2,937
Lines of Code in KLOC	----------	--------	----------	----------
Mode ran in	Semidetached	Embedded	Organic	Semidetached
Employee Salary per month	9476	7500	8000	9476

COCOMO Input	Excel	Power Point	MSRT	Java Script OS
Executable size in kb	25,100	1,806	136,055	----------
Lines of Code in KLOC	----------	----------	----------	6
Mode ran in	Embedded	Organic	Embedded	Organic
Employee Salary per month	10,000	10,000	10,000	0

4.2 COCOMO Output

COCOMO Output	Hearthstone	Steam	Notepad++	Battlenet
M Coefficient	0.4582	.5679	1	0.4582
R Coefficient	2.5	2.5	2.5	2.5
S Coefficient	.32	.35	0.38	0.32
Lines of Code	559,404	106,863	71,625.2	104,409
Kilo Lines of Code	559.404	106.863	71.6252	104.409
Effort	1,643.12	432.564	212.829	250.729
Productivity	340.453	247.045	336.539	416.423
Time	26.7272	20.9198	19.1686	14.6449
Staff needed	61	20	11	17
Cost in dollars	12,600,000	3,200,000	1,700,000	2,300,000

COCOMO Output	Excel	PowerPoint	MSRT	JS OS
M Coefficient	0.26	0.3015	0.8472	0.7147
R Coefficient	2.5	2.5	2.5	2.5
S Coefficient	0.32	0.38	0.35	0.38
Lines of Code	892,472	64,193.6	4,837,760	6000
Kilo Lines of Code	892.472	64.1936	4,837.76	6
Effort	1,594.53	76.2816	62,624.1	15.009
Productivity	559.709	841.535	77.2509	399.761
Time	26.4717	13	119.344	7
Staff needed	60	5	524	2
Cost in dollars	15,000,000	762,816	620,000,000	0

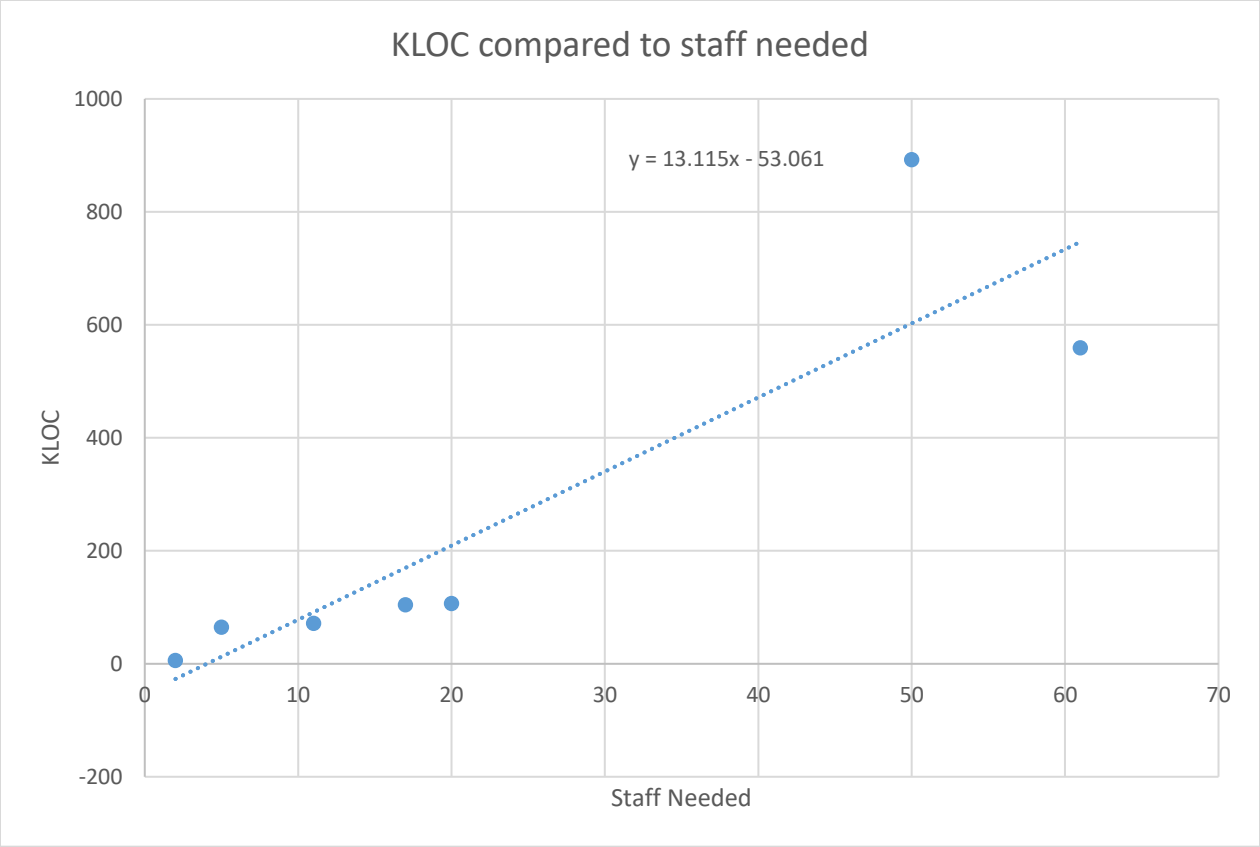

Figure 1

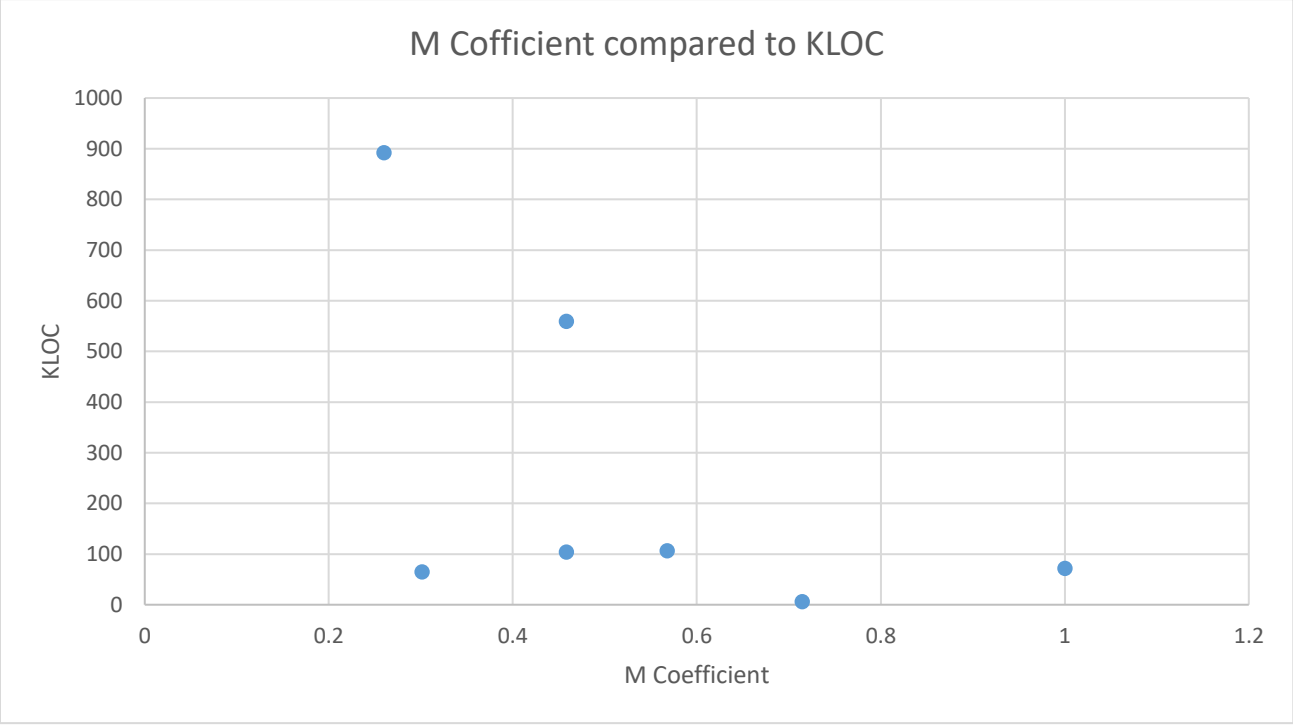

Figure 2

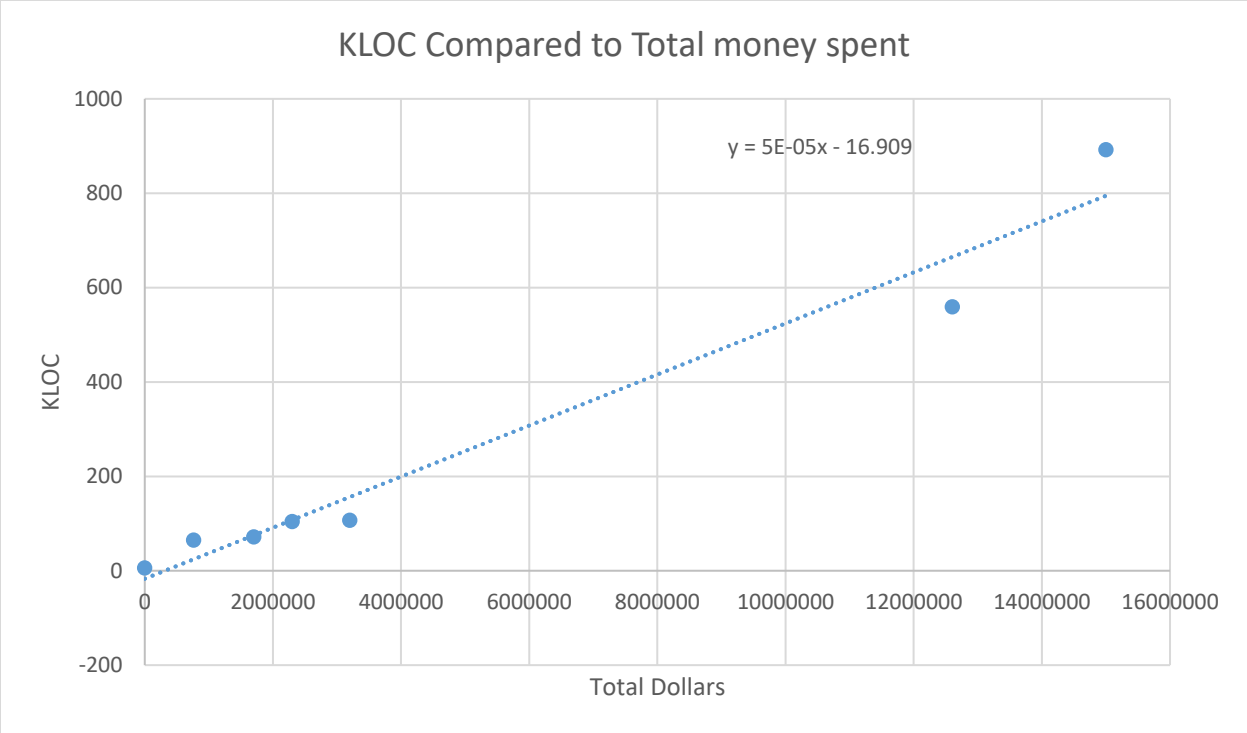

Figure 3

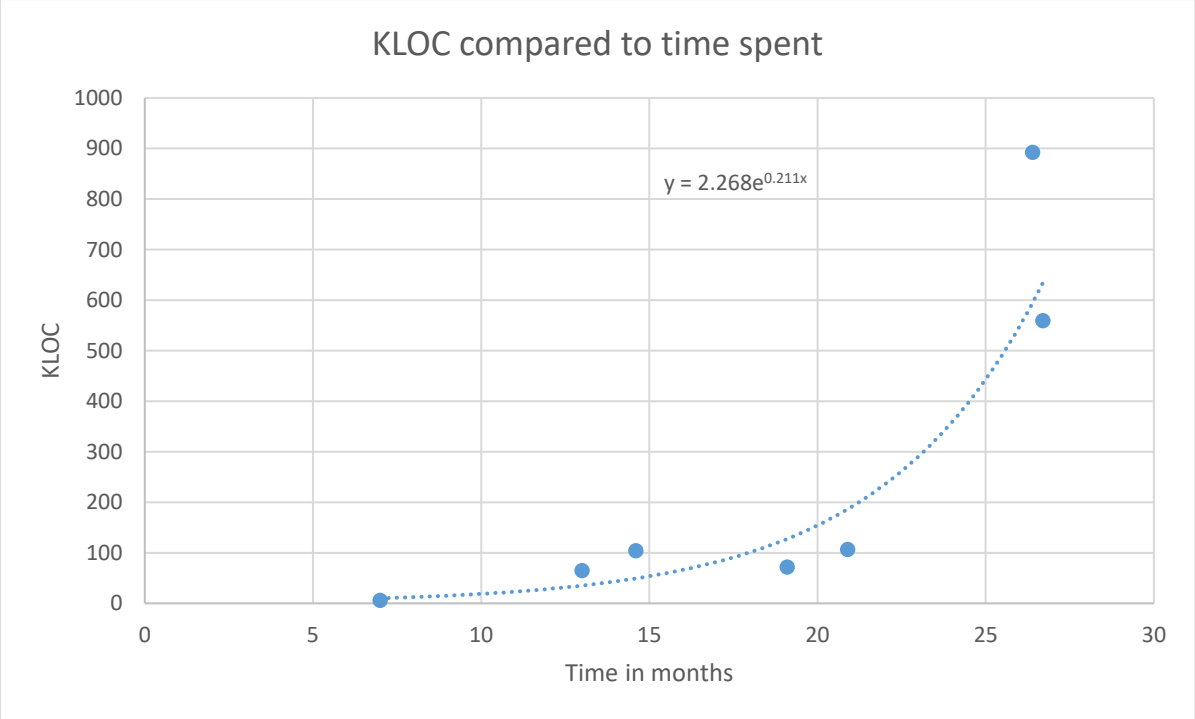

Figure 4

5.0 Data analysis

First I'd like to talk about the input for the constructive cost model. The cost per engineer is actually accurate as I went on to GlassDoor.com and found the average software engineer salary. Dividing this by 12 you can get the monthly cost of an engineer, take note you do not take into account government taxes. You don't take into account taxes, because this should be money fronted by the company as a cost for an engineer not how much an engineer makes after taxes. The java script OS project is a school project in CSC 415, so naturally the cost per month of each engineer is 0.

The executable size is grabbed straight from my computer as I have all of these applications currently installed. The java script OS input is in KLOC, because we developed that ourselves and know exactly how many lines of code there are in it. For testing purposes I also ran the COCOMO program on the executable of the java script OS which was 221 kb. With this it was only off by 1000 lines of code, so I figured that was a reasonable enough error in which I could get good results.

The mode that the constructive cost model is ran in was selected on a case by case basis. For example excel being a 25,100 kb executable I gave it embedded due to its size. This is the same case with the malicious software removal tool that Microsoft made. It is a very complex tool with a lot of innovation so I thought embedded would fit it well. As for things like my java script operating system organic fits that perfectly as it is a small project, with loose guidelines. If it was a mix between those two I assigned it semidetached. For example Hearthstone is a large project; however, there is little innovation needed because it's made by Blizzard Entertainment. So naturally being a game company they have had their fair share of practice developing games so little innovation is needed.

Moving on to how I selected the drivers for each application. I did some digging on how each company works, and hires software engineers. For example a company like Google or Microsoft will probably have very talented engineers as it is very difficult to get hired at these companies. Also I made judgements on how difficult certain projects are. For example the Microsoft software removal tool is a very complex project, so I leaned more towards the very high, high side for applicable categories. Also for large companies that are very reputable I assumed that they used modern programming practices efficiently.

Comparing KLOC to staff needed we find something that has some positive relation for my sample size. As the number of lines of code increase the number of people that will be needed to work on the project are going to increase. Logically that sounds correct since you can't have one person write 300K lines of code, and oppositely you wouldn't want 20 people writing 10K lines of code. With these trend lines you can now have a better estimation of how many people you will need just by knowing how many lines of code you need to write.

Looking at the M Coefficient calculated from the drivers, and comparing that to the number of lines of code needed to be written I don't see any correlation. It appears how the group is structured does not affect the degree of work for the project at all which logically seems write. No matter how you manage your group you still need to do the same amount of work, although I think some ways are better than others and may take longer to accomplish otherwise.

Comparing KLOC to the total amount of money spent we see a positive correlation. I believe this correlation is there because as lines of code needed increases the number of staff needed also increases; therefore, the total money spent must also increase if the number of staff is increasing. Logically thinking about this it makes sense for the increase with KLOC so it appears this checks out as well.

Moving on to looking at time spent developing the application and number of lines of coded for the application. Here we see something a little different. We see that it fits the line of an exponential function quite well, but what does this mean? Well for smaller projects the complexity is usually not to the degree of something very large. However, I think we can safely assume projects with 900K lines of code have a much larger complexity than something with only 100 lines. There are probably more moving parts to the project, and more things to keep track of when adding functionality to it. So we see projects that are going to take a very long time to develop require a far greater number of lines of code.

5.0 Conclusions

Overall from this project it's interesting to see the size and scope of projects, and how they vary in terms of how long and how much it would cost to complete one. It gives me a new appreciation for when companies release software for free, because I know how much money they sink into these applications. Especially applications that I use frequently such as, Excel. Hearthstone, Steam, and notepad++. However I do wonder how effective COCOMO is when applied to certain teams. For example if a team is comprised of 2 very talented coders, but the other 4 or 5 are not very talented. I feel COCOMO falls apart at this junction, because it assumes that skill per individual is rather equal when in reality it is far from that. Also the drivers we are using are from a certain company. To get very good results with little to zero error, we would have to develop drivers from the company we work at. This however is not as easy as it sounds as you need hundreds of projects to gauge how well or how poorly the drivers should effect the numbers. I do believe though that the general theme of the numbers would remain true throughout no matter where you work.

6.0 Appendix

A.) Source code

```
//Name: Matthew Wishoff
//Date: 3/29/16
//Class: CSC 840
//Description: Write a program that evaluates the COCOMO model of cost analysis.

//The outputs should be the team size, the program development time, the
//total development cost, etc. Your program should include cost drivers.

#include <iostream>
#include <fstream>
#include <stdio.h>
#include <math.h>

using namespace std;
//cout << driverNames[driverNumber] + "\t";
//Put length for in for 15 instead?
#define DisplayDriver(driverNumber)
                                                \
                printf("%-36s", driverNames[driverNumber].c_str());
                        \
                for(j = 1; j <= 6; j++)
                                                        \
                {
                                                                                                \
                        if(m##driverNumber[j] != 0.00){
                                                        \
                                switch(j){
                                                                                \
```

```
                case 1:                                                 \
                        cout << " [1] Very Low ";                       \
                        break;                                          \
                case 2:                                                 \
                        cout << " [2] Low ";                            \
                        break;                                          \
                case 3:                                                 \
                        cout << " [3] Normal ";                         \
                        break;                                          \
                case 4:                                                 \
                        cout << " [4] High ";                           \
                        break;                                          \
                case 5:                                                 \
                        cout << " [5] Very High ";                      \
                        break;                                          \
                case 6:                                                 \
                        cout << " [6] Extra High ";                     \
                        break;                                          \
        }                                                               \
}                                                                       \
else{                                                                   \
        switch(j){                                                      \
                case 1:                                                 \
                        cout << "          ";                          \
                        break;                                          \
                case 2:                                                 \
                        cout << "       ";                             \
                        break;                                          \
                case 3:                                                 \
```

```
                                        cout << "          ";
                                        \
                                        break;
                                                                \
                                case 4:
                                                                \
                                        cout << "        ";
                                        \
                                        break;
                                                                \
                                case 5:
                                                                \
                                        cout << "        ";
                                        \
                                        break;
                                                                \
                                case 6:
                                                                \
                                        cout << "            ";
                                \
                                        break;
                                                                \
                        }
                                                                        \
                }
                                                                        \
        }
                                                                        \
        cout << endl;
                                                                \
        cout <<   "Enter the corresponding number: ";
                        \
        flag = true;
                                                                \
        while(flag)
                                                                \
        {
                                                                        \
                cin >> driverSetting;
                                                        \
                if(m##driverNumber[driverSetting] >= 0.70 &&
                \
                  m##driverNumber[driverSetting] <= 1.66 &&
                \
                  driverSetting <= 6 &&
                                        \
                  driverSetting >= 1   ){
                                        \
                        selectedDrivers[driverNumber - 1] = m##driverNumber[driverSetting];
\
                        flag = false;
                                                                \
                }
                                                                        \
                else{
                                                                \
                        cout << "Invalid selection" << endl;
                        \
```

```
                                        cout << "Enter the corresponding number: ";
                                        \
                                }
                                                                                                \
                        }
                                                                                                \
                        cout << endl;


#define Display(method)

        \
                int j;

                        \
                DisplayDriver(1); DisplayDriver(2); DisplayDriver(3); DisplayDriver(4); DisplayDriver(5); DisplayDriver(6);
        \
                DisplayDriver(7); DisplayDriver(8); DisplayDriver(9); DisplayDriver(10); DisplayDriver(11); DisplayDriver(12);
        \
                DisplayDriver(13); DisplayDriver(14); DisplayDriver(15); if(method == 16){ DisplayDriver(method); }

#define thousand 1000

int main()
{
        double kloc;                                            // The number of logical lines of code in the program.
        int memSize;                                    // The memory size of the program.
        int personCost;                                 // Average cost of employees working on the project.
        char hasKLOC;                                   // Yes or no value if the user has the KLOC of the program.
        char hasMemSize;                        // Yes or no value if the user has the memory size of program.
        string model;                                   // Which model the program is going to be using to determine the
output.
        string driverModel;                     // takes in which driver model you would like to use.
        bool flag = true;                       // flag to keep while loop executing until parameters are met.
        double aCoefficient;                            // Value assigned depending on which cocomo model you choose
        double bCoefficient;                            // Value assigned depending on which cocomo model you choose
        double mCoefficient = 1.00;     // Value assigned depending on which drivers are used...
        double rCoefficient;
        double sCoefficient;
        int driverSetting;                      // Value used to change the value of the driver.
        double effort;                                  //
        double productivity;
        double time;
        int staffNum;
        double cost;
        double loc;                                             // loc == 1000 * kloc

        /*
                Drivers
                very low = 1
                low = 2
                nominal = 3
                high = 4
                very high = 5
                extra high = 6
                positions in the drivers that have value 0.00 are not applicable.
        */
```

```
		//Product attributes
		double m1[7] = {0.00, 0.75, 0.88, 1.00, 1.15, 1.40, 0.00}; // Required software reliability
		double m2[7] = {0.00, 0.00, 0.94, 1.00, 1.08, 1.16, 0.00}; // Data base size
		double m3[7] = {0.00, 0.70, 0.85, 1.00, 1.15, 1.30, 1.65}; // Product complexity

		// Computer Attributes
		double m4[7] = {0.00, 0.00, 0.00, 1.00, 1.11, 1.30, 1.66}; // Execution time constraint
		double m5[7] = {0.00, 0.00, 0.00, 1.00, 1.06, 1.21, 1.56}; // Main storage constraint
		double m6[7] = {0.00, 0.00, 0.87, 1.00, 1.15, 1.30, 0.00}; // Virtual machine volatility
		double m7[7] = {0.00, 0.00, 0.87, 1.00, 1.07, 1.15, 0.00}; // Computer turnaround time

		//Peronnel Attributes
		double m8[7] = {0.00, 1.46, 1.19, 1.00, 0.86, 0.71, 0.00}; // Analyst capabilities
		double m9[7] = {0.00, 1.29, 1.13, 1.00, 0.91, 0.82, 0.00}; // Applications experience
		double m10[7] = {0.00, 1.42, 1.17, 1.00, 0.86, 0.70, 0.00}; // Programmer capability
		double m11[7] = {0.00, 1.21, 1.10, 1.00, 0.90, 0.00, 0.00}; // Virtual machine experience
		double m12[7] = {0.00, 1.14, 1.07, 1.00, 0.85, 0.00, 0.00}; // Programming language experience

		//Project Attributes
		double m13[7] = {0.00, 1.24, 1.10, 1.00, 0.91, 0.82, 0.00}; // Use of modern programming practices
		double m14[7] = {0.00, 1.24, 1.10, 1.00, 0.91, 0.83, 0.00}; // Use of software tools
		double m15[7] = {0.00, 1.23, 1.08, 1.00, 1.04, 1.10, 0.00}; // Required develoopment schedule
		double m16[7] = {0.00, 0.00, 0.91, 1.00, 1.09, 1.35, 1.62}; // Requirements volatility

//		double drivers[16] = { m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15, m16 };
		string driverNames[17] = {" ", "software reliability", "Data base size", "Product complexity", "Execution time
constraint",
                                                                "Main storage constraint", "Virtual machine volatility",
"Computer turnaround time", "Analyst capabilities",
                                                                "Applications experience", "Programmer capability",
"Virtual machine experience", "Programming language experience",
                                                                "Use of modern programming practices", "Use of
software tools", "Required development schedule", "Requirements volatility"};
		double selectedDrivers[16] = {};
		for(int i = 0; i <= 15; i++)
		{
			selectedDrivers[i] = 1.00;
		}

		//Continue looping until the parameters of the program have been met (kloc or memSize).
		while(flag)
		{
			cout << "Do you have KLOC? (y/n):  ";
			cin >> hasKLOC;
			cout << "Do you have the memory size? (y/n):  ";
			cin >> hasMemSize;
			cout << "Enter the cost per person per month ($/month):  $";
			cin >> personCost;

			cout << endl;

			if(hasKLOC == 'y')
			{
				cout << "Enter the size of the source program (KLOC):  ";
				cin >> kloc;
				flag = false;
			}
```

```
                    if(hasMemSize == 'y')
                    {
                              cout << "Enter the size of occupied memory (KiloBytes):  ";
                              cin >> memSize;
                              cout << "memSize before: " << memSize << endl;
//                            loc = memSize/30.8; (This formula is wrong the one below is explained below.
                              // y = memsize, and x = loc
                              //y = 40.371x - 656.33
                              //memsize = 40.371(loc) - 656.33
                              //memsize + 656.33 = 40.371(loc)
                              //(memsize + 656.33)/40.371 = loc
                              memSize *= 1000;
                              kloc = ((memSize - 656.33) / (28.1234 * 1000));
                              loc = kloc*1000;
                              cout << "memsize: " << memSize << endl;
                              cout << "kloc: " << kloc << endl;
                              cout << "loc: " << loc << endl;
//                            loc = ((memSize + 656.33)/40.371);

//                            kloc = loc * thousand;

                              //Convert memsize to lines of code,
                              //Convert lines of code to KLOC.

                              flag = false;
                    }

                    if(hasMemSize == 'n' && hasKLOC == 'n')
                    {
                              cout << "You do not have enough valid information to compute cost analysis." << endl;
                              cout << "Please enter at least the KLOC or Memory size of the program to continue . . . " << endl;
                              cout <<
"********************************************************************************** " << endl << endl;
                    }
          }
          cout << endl;

          cout << "********************************************************************************** " << endl;

                    cout <<  "\t \t  Pick the driver that best fits your project" << endl;
                    cout << "basic: " << endl;
                    cout << "\t The m multiplier will be set to the value 1.00" << endl << endl;
                    cout << "medium: " << endl;
                    cout << "\t The m multiplier will be made up of 15 different elements" << endl;
                    cout << "\t the default of these elements is set to 1.00, and can be" << endl;
                    cout << "\t modified individually." << endl << endl;
                    cout << "advanced: " << endl;
                    cout << "\t The m multiplier will be made up of 16 different elements" << endl;
                    cout << "\t the default of these elements is set to 1.00, and can be" << endl;
                    cout << "\t modified individually." << endl << endl;

          flag = true;
          while(flag)
          {
                    cout << "Enter the driver mode you would like (basic, medium, advanced): ";
                    cin >> driverModel;
                    cout << endl;
```

```
		if(driverModel == "basic")
		{
			cout << "Entering " + driverModel + " driver mode . . ." << endl;
			flag = false;
		}
		else if(driverModel == "medium")
		{
			cout << "Entering " + driverModel + " driver mode . . ." << endl << endl;
			Display(15); //Display M-drivers to modify
			flag = false;
		}
		else if(driverModel == "advanced")
		{
			cout << "Entering " + driverModel + " driver mode . . ." << endl << endl;
			Display(16); //Display M-drivers to modify
			flag = false;
		}
		else
		{
			cout << "invalid driver model entered." << endl;
		}
}
for(int i = 0; i <= 15; i++)
{
		mCoefficient = mCoefficient * selectedDrivers[i];
}
cout << "mCoefficient: " << mCoefficient << endl;

cout << endl;
cout << "******************************************************************************** " << endl;

cout << "\t \t  Pick the mode that best describes your project" << endl;
cout << "Organic: " << endl;
cout << "\t relatively small size, little innovation, relaxed delivery" << endl;
cout << "\t requirements, development in a stable in-house environment" << endl << endl;

cout << "Embedded: " << endl;
cout << "\t relatively large, operating within tight constraints, greater" << endl;
cout << "\t innovation needs, high hardware and customer interface complexity," << endl;
cout << "\t rigid requirements." << endl << endl;

cout << "Semidetached: " << endl;
cout << "\t located in between the organic and embedded modes." << endl << endl;

flag = true;
while(flag)
{
		cout << "Enter Organic, Embedded, or Semidetached: ";
		cin >> model;
		cout << endl;

		if(model == "Organic" || model == "organic")
		{
			flag = false;
			cout << "Entering " + model + " mode . . ." << endl;
			if(driverModel == "basic"){
				aCoefficient =2.40;
				bCoefficient =1.05;
```

```
		}
		else if(driverModel == "medium"){
			aCoefficient =3.20;
			bCoefficient =1.05;
		}
		else if(driverModel == "advanced"){
			aCoefficient =2.60;
			bCoefficient =1.08;
		}
		rCoefficient = 2.5;
		sCoefficient = 0.38;

	}
	else if(model == "Embedded" || model == "embedded")
	{
		flag = false;
		cout << "Entering " + model + " mode . . ." << endl;
		if(driverModel == "basic"){
			aCoefficient =3.60;
			bCoefficient =1.20;
		}
		else if(driverModel == "medium"){
			aCoefficient =2.80;
			bCoefficient =1.20;
		}
		else if(driverModel == "advanced"){
			aCoefficient =2.90;
			bCoefficient =1.20;
		}
		rCoefficient = 2.5;
		sCoefficient = 0.35;

	}
	else if(model == "Semidetached" || model == "semidetached")
	{
		flag = false;
		cout << "Entering " + model + " mode . . ." << endl;
		if(driverModel == "basic"){
			aCoefficient =3.00;
			bCoefficient =1.12;
		}
		else if(driverModel == "medium"){
			aCoefficient =3.00;
			bCoefficient =1.12;
		}
		else if(driverModel == "advanced"){
			aCoefficient =2.90;
			bCoefficient =1.12;
		}
		rCoefficient = 2.5;
		sCoefficient = 0.32;

	}
	else
	{
		cout << endl << endl;
		cout << "Incorrect model entered." << endl;
	}
```

```
        }

        cout << "******************************************************************************** " << endl;
        cout << "\t \t Calculated Values" << endl;
        loc = kloc * thousand;
        effort = aCoefficient * pow(kloc, bCoefficient) * mCoefficient;
//      effort = aCoefficient * loc^bCoefficient * mCoefficient;
        productivity = loc / effort;
        time = rCoefficient * pow(effort, sCoefficient);
        staffNum = effort / time;
        cost = personCost * effort;

        cout << "aCoefficient = " << aCoefficient << endl;
        cout << "bCoefficient = " << bCoefficient << endl;
        cout << "mCoefficient = " << mCoefficient << endl;
        cout << "rCoefficient = " << rCoefficient << endl;
        cout << "sCoefficient = " << sCoefficient << endl;
        cout << "Loc         = " << loc << endl;
        cout << "kloc        = " << kloc << endl << endl;

        cout << "******************************************************************************** " << endl;
        cout << "\t\t End Results" << endl;


        cout << "Effort      = " << effort << endl;
        cout << "Productivity = " << productivity << endl;
        cout << "Time        = " << time << endl;
        cout << "Staff needed = " << staffNum << endl;



        cout << "Cost        = " << cost << endl;

}
```

B.)Example Output

```
Do you have KLOC? (y/n):  n
Do you have the memory size? (y/n):  y
Enter the cost per person per month ($/month):  $8000

Enter the size of occupied memory (KiloBytes):  15766
memSize before: 15766
memsize: 15766000
kloc: 560.577
loc: 560577

*****************************************************************************
            Pick the driver that best fits your project
basic:
      The m multiplier will be set to the value 1.00

medium:
      The m multiplier will be made up of 15 different elements
      the default of these elements is set to 1.00, and can be
      modified individually.

advanced:
      The m multiplier will be made up of 16 different elements
      the default of these elements is set to 1.00, and can be
      modified individually.

Enter the driver mode you would like (basic, medium, advanced): medium

Entering medium driver mode . . .

software reliability              [1] Very Low  [2] Low  [3] Normal  [4] High  [5] Very High
Enter the corresponding number: 5

Data base size                          [2] Low  [3] Normal  [4] High  [5] Very High
Enter the corresponding number: 2

Product complexity                [1] Very Low  [2] Low  [3] Normal  [4] High  [5] Very High  [6] Extra High
Enter the corresponding number: 4

Execution time constraint                       [3] Normal  [4] High  [5] Very High  [6] Extra High
Enter the corresponding number: 4

Main storage constraint                         [3] Normal  [4] High  [5] Very High  [6] Extra High
Enter the corresponding number: 4

Virtual machine volatility                [2] Low  [3] Normal  [4] High  [5] Very High
Enter the corresponding number: 3
```

Computer turnaround time [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 4

Analyst capabilities [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 4

Applications experience [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 5

Programmer capability [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 5

Virtual machine experience [1] Very Low [2] Low [3] Normal [4] High
Enter the corresponding number: 3

Programming language experience [1] Very Low [2] Low [3] Normal [4] High
Enter the corresponding number: 4

Use of modern programming practices [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 5

Use of software tools [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 5

Required development schedule [1] Very Low [2] Low [3] Normal [4] High [5] Very High
Enter the corresponding number: 1

mCoefficient: 0.669257

****

Pick the mode that best describes your project

Organic:
relatively small size, little innovation, relaxed delivery
requirements, development in a stable in-house environment

Embedded:
relatively large, operating within tight constraints, greater
innovation needs, high hardware and customer interface complexity,
rigid requirements.

Semidetached:
located in between the organic and embedded modes.

Enter Organic, Embedded, or Semidetached: Organic

Entering Organic mode . . .
****

Calculated Values
aCoefficient = 3.2
bCoefficient = 1.05
mCoefficient = 0.669257
rCoefficient = 2.5
sCoefficient = 0.38
Loc = 560577
kloc = 560.577

****

End Results
Effort = 1647.44
Productivity = 340.271
Time = 41.7184
Staff needed = 39
Cost = 1.31795e+007

Process exited after 65.36 seconds with return value 0
Press any key to continue . . .