Variation in Solving Programming Problems in an Online SUNY Learning Network Course

Alireza Ebrahimi, Ph.D.

MIS - School of Business

SUNY College at Old Westbury

An observation on teaching introductory programming courses on SLN for a period of two terms led me to believe that on-line students try various ways to solve a problem. In the beginning, I got the impression that some of their solution approaches were wrong; but after a little investigation, I found that some of the problems solving approaches were correct but difficult to understand and not straight forward. However, some of the solutions were interesting, challenging, and unique. Three problem sets will be discussed which were given from the textbook in modules 2 and 3 of the term. The problems were to compute salary with overtime, compute total average mileage per gallon, and parking charge fee. As an instructor who has taught introductory programming for more than two decades, I found several variations in solving a problem, even in a small simple program like a compute salary program. One speculation would be that students use trial and error and their own logic to get the correct output, despite not using a common approach. It seems that students in the face to face classes focus on the information from the instructor and disseminate it among each other.

Introduction and background studies

As online learning becomes a common tool many researchers are trying to compare the performance in online and traditional courses and how are they different. In a study of online and traditional MIS students by Gary Ury indicate "The online delivery method was found to be effective, but performance, as measured by final course grades, showed a significantly lower mean score than students enrolled in traditional sections of the course (Ury, 2004)". In analysis of student data by Kleinman and Entin support there is know significant difference between online and traditional teaching. Technology difficulties were responsible for a large drop out of online students with in the first few weeks of the semester. Those students that completed the course were more positive about the value of the course (Kleinman, 2002). In this study contrast does not investigate whether online student perform better than traditional students, rather online students solve problems in a different ways that some are not understandable and common.

Observation and Initial Impression: Output is Correct, Therefore Student's Program is Correct and Professor is Wrong

I have made several observations while teaching introductory programming courses on SLN for a period of two terms (Fall 2005 and Summer 2006). There are 20 students taking online Introduction to C++ Programming and OOP at Empire State College SUNY Learning Network (SLN). There were no programming prerequisites for this course. While working on each module, students can ask questions on the subject matter, do their own research, and study given reading material. Students will write their programs, test, and submit the working version with the test data. As an instructor, I evaluate a program by looking at the algorithm, the output, and checking for possible errors. A written comment and a grade will be given for each module assignment.

My investigation led me to believe that on-line students try various ways to solve a problem. In the beginning, I got the impression that some of their solution approaches were wrong but after a little investigation, I found that solution was correct and some of the problem solving approaches was difficult to understand, complicated and not straightforward. However some of the solutions were Unique, challenging and interesting.

Student Problem Set (Salary with Overtime, Average Mileage per Gallon, and Parking Charge Fee)

Students were asked to solve three problems as their assignments on the third week (module 2) and the fifth week (module 3) of the course. The first assignment is known as salary with overtime pay. The input for salary program was the amount of hours worked and hourly rate. The second problem was to compute average mileage per gallon (MPG). The input for the MPG program was the amount of mileage and amount of gallons. Students need to find average mileage per gallon for all cars entered. The programming topic for the first and second assignment are control structures such as if statement, and loops, as well as simple arithmetic and assignment statements. A dummy value known as sentinel (-1) is used to stop input entries. Students are warned to prompt and print an error message for divide by zero when calculating the average.

The third assignment is known as parking charge fee. The inputs were the amount of hours each car parked in the garage. The parking charge fee was computed for three cars and displayed in a tabular format. The programming topic for the third assignment was on functions and how a function returns a value to be used in a main program. This set of problems was taken from the course textbook (Deitel, 2005). To demonstrate how online students solve problem differently from traditional face to face students here 6 samples is from 90 samples of the written students programs.

How Do Online Students Solve the Payroll Program Differently from Traditional Students?

In the payroll program, the user will compute the salary of an hourly-based employee with overtime pay. The user enters the amount of hours worked and hourly rate. If an employee works more than 40 hours they will receive overtime pay of one and a half time hourly rate for each hour over 40. There were many variations in students' programs to solve this problem. For example, for an employee who worked 50 hours with an hourly rate of 10, the gross pay salary can be computed as follows:

(40 * hourlyrate) + 10 * hourlyrate * 1.5

Traditional face to face students learning programming will solve the salary program with overtime pay by calculating overtime hour pay and regular pay as shown in the following program (Ebrahimi, 2002):

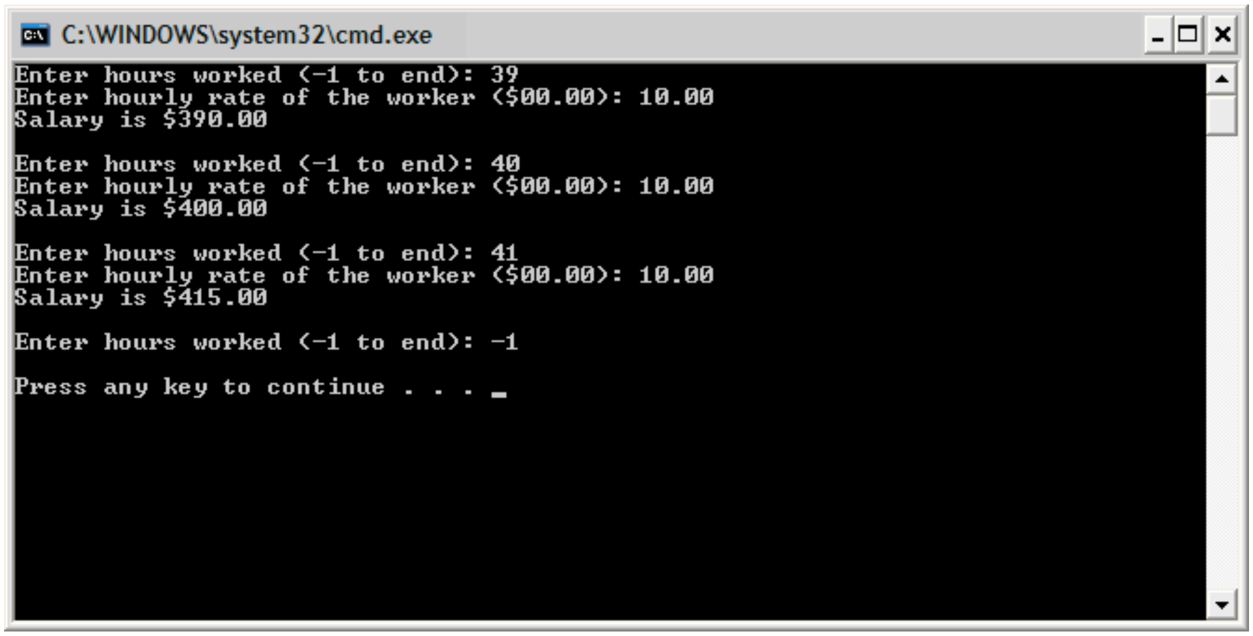
```
#include<iostream>
using namespacestd;
main(){
      long int empid;
      int hoursworked, overtimehours;
      float hourlyrate, regularpay,overtimepay, grosspay ;
      while( cin>>empid>>hoursworked>>hourlyrate ){
           if( hoursworked>40){
                overtimehours = hoursworked -40;
                overtimepay = overtimehours *hourlyrate * 1.5;
                regularpay = 40 *hourlyrate;
           }//IF
           else{
                overtimehours =0;
                overtimepay = 0;
                regularpay = hoursworked *hourlyrate;
           }//ELSE
           grosspay = regularpay +overtimepay;
           cout<<" EMPLOYEE ID   IS "<<empid<<endl;
           cout<<" OVERTIME PAY IS "<<overtimepay<<endl;
           cout<<" GROSS PAY        IS "<<grosspay<<endl;
      }//WHILE
      return 0;
}//MAIN
```

The following are some of the variations that online students will have while programming salary with overtime pay:

In variation 1 student changes hours, variation 2 changes the rate and variation 3 compute overtime pay for every employee and only display the one that works overtime.

Variation 1 (ID Name = RD)

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
using std::fixed;
#include <iomanip>
using std::setprecision;           // for a 2 place decimal
int main()                         // Function main
{
double hours = 0;             // initialize variable to zero
double rate = 0;              // initialize variable to zero
double salary = 0;            // initialize variable to zero
 cout << "Enter hours worked (-1 to end): ";
 cin >> hours;          // prompt for hours or -1
 while ( hours != -1 ) {       // if input != -1 end while loop
 cout << "Enter hourly rate of the worker ($00.00): ";
 cin >> rate;
 if ( hours > 40 )       // Test for overtime & calculate hours
```
hours = ((hours - 40) * 1.5) + 40;
salary = (rate * hours);
```
        cout << "Salary is "
             << setprecision(2) << fixed << "$" << salary << "\n\n";
            cout << "Enter hours worked (-1 to end): ";
        cin >> hours;
    }
cout << endl;
    return 0;
}
```



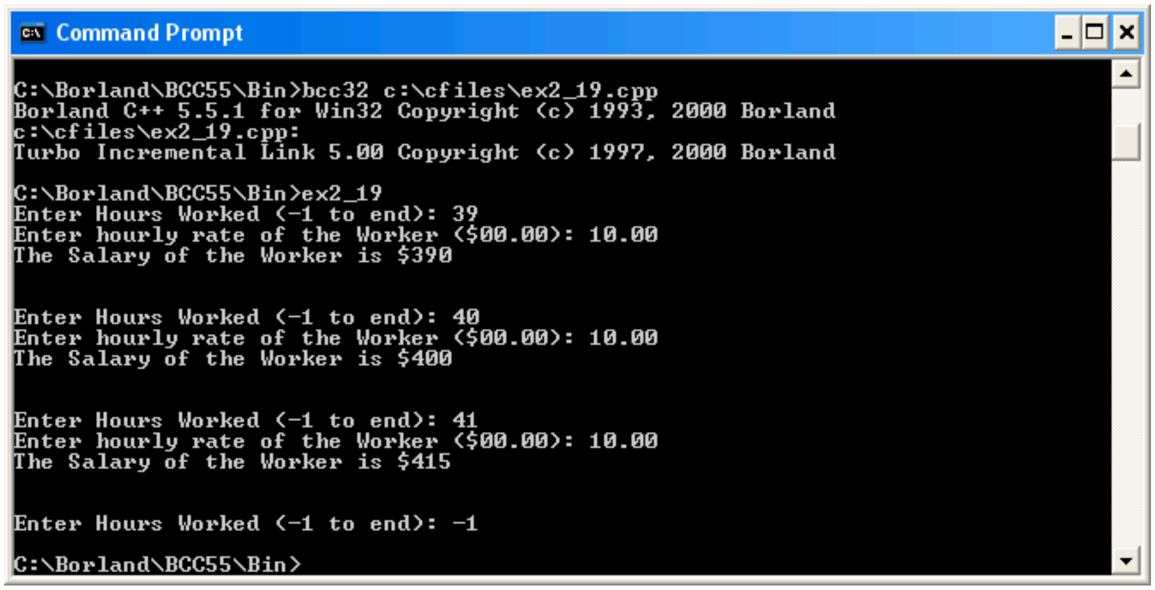
```
Enter hours worked (-1 to end): 39
Enter hourly rate of the worker ($00.00): 10.00
Salary is $390.00

Enter hours worked (-1 to end): 40
Enter hourly rate of the worker ($00.00): 10.00
Salary is $400.00

Enter hours worked (-1 to end): 41
Enter hourly rate of the worker ($00.00): 10.00
Salary is $415.00

Enter hours worked (-1 to end): -1

Press any key to continue . . .
```

In computing the salary program the student (ID Name = RD) is figuring out the amount of overtime hours in term of regular hour by multiplying by 1.5.

```
        if ( hours > 40 )       // Test for overtime & calculate hours
        hours = ((hours - 40) * 1.5) + 40;
        salary = (rate * hours);
```

Variation 2(ID Name = CR)

```
#include <iostream>
using std :: cin ;
using std  :: cout ;
int main()
{
          int hours;
          double rate;
          double salary;

cout<<"Enter Hours Worked (-1 to end): ";
cin>>hours;
          while(hours!=-1){
                    cout<<"Enter hourly rate of the Worker

                    ($00.00): ";
                    cin>>rate;
                    if(hours>40){
                              salary=40*rate;
                              hours=hours-40;
                              rate=rate+rate/2;
                              salary+=(rate*hours);
                    }
                    else{
                              salary=rate*hours;
                    }
                    cout<<"The Salary of the Worker is $";
                    cout<<salary;
                    cout<<"\n";
                    cout<<"\n\nEnter Hours Worked (-1 to end):

";
                    cin>>hours;
          }
          return 0;

}
```



```
C:\Borland\BCC55\Bin>bcc32 c:\cfiles\ex2_19.cpp
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
c:\cfiles\ex2_19.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

C:\Borland\BCC55\Bin>ex2_19
Enter Hours Worked (-1 to end): 39
Enter hourly rate of the Worker ($00.00): 10.00
The Salary of the Worker is $390


Enter Hours Worked (-1 to end): 40
Enter hourly rate of the Worker ($00.00): 10.00
The Salary of the Worker is $400


Enter Hours Worked (-1 to end): 41
Enter hourly rate of the Worker ($00.00): 10.00
The Salary of the Worker is $415


Enter Hours Worked (-1 to end): -1

C:\Borland\BCC55\Bin>
```

In computing the salary program the student (ID Name = CR) is figuring out the amount of overtime by changing the rate.

```
if(hours>40){
        salary=40*rate;
        hours=hours-40;
        rate=rate+rate/2;
        salary+=(rate*hours);
}
  else{
        salary=rate*hours;
```

In variation 3(ID Name = FK) the student compute overtime pay for ever salary however the program will not display salary overtime pay if hours worked is not greater than 40.

Variation 3 (ID = FK)

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
int main()
{
double hours;
double rate;
double pay;
double othours;
double otsal;
double otpay;
double salary;
cout << "Enter hours worked (-1 to end): ";
cin >> hours;
while ( hours != -1 ){
cout << "Enter hourly rate of the worker ($00.00): ";
cin >> rate;
pay = hours * rate;
otsal = hours * rate;
otpay = (otsal / hours) * 1.5;
othours = 40 * rate;
salary = otpay + othours;
if ( hours <= 40 )
cout << "Salary is $" << pay << endl;
else
cout << "Salary is $" << salary << endl;
cout << "\nEnter hours worked (-1 to end): ";
cin >> hours;
}
return 0;}
```

How Do Online Students Solve the Mileage-Per-Gallon Program Differently from Traditional Students?

In the second exercise the online students were asked to find Mile-Per-Gallon (MPG) for every car, Average Mile-Per-Gallon for all cars and display MPG and total MPG. A common solution by traditional face to face student will be to calculate Mileage per Gallon for each car (MPG), accumulate each MPG and count the number of cars. To find the total average is computed by dividing sum of the MPG by the total number of the cars. In term of division my zero a face to face student will check the car number against zero. This the common code for the face to face students:

```
{mpg = mileage/gallon

totmpg = totmpg  + mpg

cars = cars + 1

}

If (car > 0)  avgmpg = totmpg/ cars

else  cout<<"errors divide by zero"
```

In a solution by online students the MPG is computed by dividing of mileage by gallon for each car the total mileage and total gallon as been computed. The total average of MPG is computed by dividing total mileage by total gallons. In case of resolving division by zero, instead of checking the total gallon other computation is checked such as average MPG of a car is checked. In other words instead of directly testing the denominator they are testing the situation prior that would lead to it.

In variation 1 this student calculated and displayed MPG for each car. The total mileage and total gallons has been computed and the average total MPG is computed my dividing the total mileage by the total gallons. Student in this variation does not handle division by zero and there is a chance of entering mileage as zero which is going to cause a problem.

Variation 1 (ID Name = SP)

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
// function main begins program execution
int main()
{
double totalg; // sum of gallons input by user
double totalm; // sum of miles input by user
double gallons; // gallons value
double miles; // miles value
double average; // average of total gallons
double avg; // average of gallons per use
int Counter; // number to be entered next
// initialization phase
totalg = 0; // initialize total gallons
totalm = 0; // initialize total miles
Counter = 1; // initialize loop counter
// processing phase
while ( Counter <= 3 ) { // loop 3 times
cout << "Enter gallons: "; // prompt for input
cin >> gallons; // read gallons from user
cout << "Enter miles: "; // prompt for input
cin >> miles; // read miles from user
avg = miles/gallons; // integer division
cout <<"The miles/ gallon for this tank was " <<avg << endl;

totalg = totalg + gallons; // add gallons to totalg
totalm = totalm + miles; // add miles to totalm
Counter = Counter + 1; // increment counter
}
// termination phase
average = totalm / totalg; // integer division
// display result
cout << "The overall averages miles/gallons was " << average << endl;
cout << "The total miles was " << totalm << endl;
cout << "The total gallons was " << totalg << endl;
return 0; // indicate program ended successfully
}
```

In this variation 2 the student calculated the MPG for each car and named average. In addition the student found the total mileage and total gallon. In case of dividing by zero for total MPG and the pervious average has been tested.

Variation 2 (ID Name = RD)

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
using std::fixed;
#include <iomanip>
using std::setprecision;                    // for a 6 place decimal
```

```
int main()                                          // Function main
{
        double gallons = 0;                 // initialize variable to zero
        double miles = 0;                   // initialize variable to zero
        double milesTotal = 0;              // initialize variable to zero
        double gallonsTotal = 0;    // initialize variable to zero
        double average = 0;                 // initialize variable to zero
                cout << "Enter the gallons used (-1 to end): ";
                cin >> gallons;             // prompt for first input value
        while ( gallons != -1 ) {   // if input value is -1 finish while loop
                gallonsTotal = gallonsTotal + gallons;
                cout << "Enter the miles driven: ";
                cin >> miles;
                milesTotal = milesTotal + miles;
                if ( gallons > 0)
                average = miles / gallons;
                cout << "The miles / gallon for this tank was "
                        << setprecision(6) << fixed << average << "\n\n";
                cout << "Enter the gallons used (-1 to end): ";
                cin >> gallons;
        }
        if ( average > 0 )          // Tests for divide-by-zero error
                        average = milesTotal / gallonsTotal;
        cout << "\nThe overall miles/gallon was " // output 6 place decimal
                << setprecision(6) << fixed << average << endl;

        return 0;
}
```

In variation 3 in the first round if its -1 it will not compute any data but how ever if the average miles-per-gallon is -1 it will not compute any data.

Variation 3 (ID Name = EW)

```
//Figures gas mileage in gallons

#include <iostream>
#include <conio.h>

using std::cout;
using std::cin;;
using std::fixed;

#include <iomanip>

using std::setprecision;
```

```
// function main begins program execution
int main()
{
        double gal;        //number of gallons used
        double miles;  //number of miles driven
        double tg;              //total of all gallons used
        double tm;     //total of all miles driven

        double mpg;                  //miles per gallon
        double total;     //average of all drives

        //initialization phase
        tg = 0;
        tm = 0;

        cout << "Enter the gallons used (Enter -1 to finish) :";
        cin >> gal;

        while ( gal != -1 )  {
        cout << "Enter the miles driven:";
        cin >> miles;

        tg = tg + gal;
        tm = tm + miles;

        //calculate miles per gallon
        mpg = static_cast < double > ( miles ) / gal;

        cout << "The miles / gallon for this tank was " << setprecision ( 6 ) << fixed <<
mpg;

        cout << "\n\nEnter the gallons used (Enter -1 to finish) :";
        cin >> gal;
        }


        ;if ( gal == -1 )
        total = static_cast < double > ( tm ) / tg;
        cout << "\nThe overall average miles/gallon was ";
        cout << total;

   getch ();
        ;return 0;

}
```

How Do Online Students Solve the Parking Charge Fee Program Differently from Traditional Students?

A parking garage problem computes a parking daily charge fee for three cars by giving the amount of hours. For the first three hours a charge fee is $2 and for any additional hours the charge fee will be 0.50. The maximum charge is $10 for a 24 hours period.

A common face to face solution is shown as below where the computation is divided into three parts the minimum charge, additional charge and a case when the charge exceeds the maximum $10.

```
 If hours<=3 charge=3;
else charge=(hours-3)* 0.50
if charge>10 charge=10.00
```

A variation case of an online solution will find the maximum hour as 16 after the first three hour or 19 including the first three hours. This is an interesting phenomenon since no where in problem set the amounts of 16 or 19 have been cited. Student will find the intermediate solution by themselves rather than relying on the program. For example since the maximum charge is $10 therefore the maximum hours that make this charge will be 16 after the first three hours.

Variation 1 (ID Name= JD)

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
#include <iomanip>
using std::setw;
```

```
double CalculatedCharge ( double ); //prototype definition for function TotalCharge
int main()
{
double hours;
double fee;
double totalCharge = 0;
double totalHours = 0;
int hourCounter = 0;
// Loop 10 times
for ( int carCounter = 1; carCounter <= 10; carCounter++)
{
// Get hours parked information from user for each car
cout << "Enter hours parked: " ;
cin >> hours;
hourCounter = hourCounter + hours;
fee = CalculatedCharge (hours);
cout << "Car " << carCounter << setw (10) << "Hours " << hours << setw (10) << fee
<<endl;
totalCharge = totalCharge + fee;
totalHours = totalHours + hours;
}
cout << "Total Hours: " << totalHours << endl;
cout << "Total Charge: " << totalCharge << endl;
return 0;
} // end main
double CalculatedCharge ( double inhours )
{
double defaultfee = 2;
double maxfee = 10;
if ( inhours <= 3){
                    return defaultfee;}
else if ( inhours >= 16) {
                           return maxfee;}
else {
              return ((( inhours -3) * .50) + 2.00);}
}
```

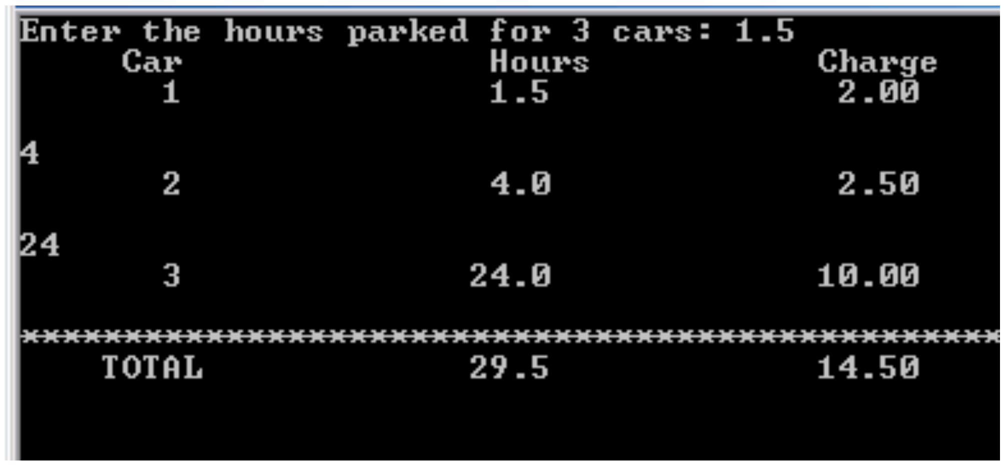
In the variation 2 of the online solution student is figuring out that 19 is the maximum number that makes the $10 charge.

Variation 1 (ID Name = GR)

```
#include <iostream>
#include <iomanip>
#include <cmath>
```

```
using namespace std;

//function prototype
double calculateCharges( double );

// //function main begins program execution
main()
{
  //declare and initialize variables
  double hour, currentCharge, totalCharges = 0.0, totalHours = 0.0;
  int header = 1;

  //prompt for user input
  cout << "Enter the hours parked for 3 cars: ";


  for ( int i = 1; i <= 3; i++ ) {
    cin >> hour;
    totalHours += hour;

      //display header
      if ( header ) {
       cout << setw( 8 ) << "Car" << setw( 20 ) << "Hours"
           << setw( 18 ) << "Charge\n";
       header = 0;   // prevents the header from printing again
    }

    //function call and assignment
      totalCharges += ( currentCharge = calculateCharges( hour ) );

      //display and format results
      cout << setiosflags( ios::fixed | ios::showpoint )
        << setw( 8 ) << i << setw( 18 ) << setprecision( 1 ) << hour
        << setw( 18 ) << setprecision( 2 ) << currentCharge << "\n\n";
  }

  cout << "********************************************************\n"
        << setw(9) << "TOTAL" << setw( 17 ) << setprecision( 1 )
      << totalHours << setw( 18 ) << setprecision( 2 )
      << totalCharges << endl;


  return 0;
}  //end function main
```

```
//Begin function
double calculateCharges( double hours )
{
  double charge;

  //calculate garage rate based on business rules
  if ( hours < 3.0 )
    charge = 2.0;
  else if ( hours < 19.0 )//after 19 hours, max charge is $10
    charge = 2.0 + .5 * ceil( hours - 3.0 );
  else
    charge = 10.0;

  return charge;//return garage rate to function call
} // end function
```



```
Enter the hours parked for 3 cars: 1.5
     Car                  Hours                Charge
       1                  1.5                   2.00

4
       2                   4.0                  2.50

24
       3                  24.0                 10.00

**********************************************
    TOTAL                 29.5                 14.50
```

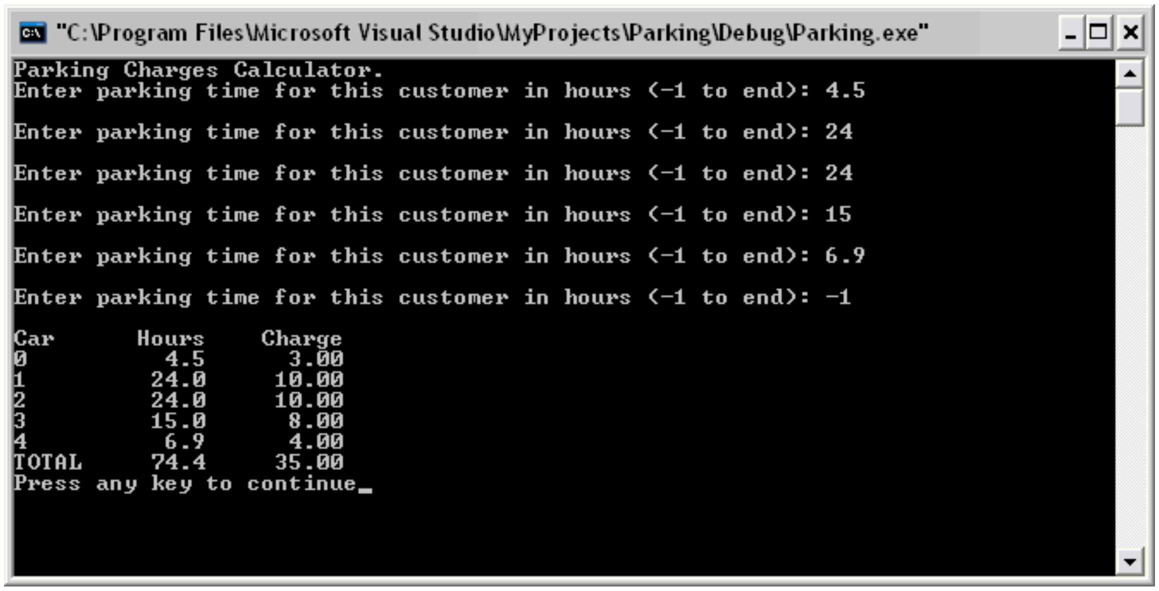
In the third online variation student solve the problem by using the concept of array which is covered in the next module. However the computation of parking charge is similar to the face to face solution.

Variation 3

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <cmath>

//Standard commands used
using std::cin;
using std::cout;
using std::endl;
using std::fixed;
using std::setprecision;
using std::setw;

// Global integers and doubles
double Hours = 0;
int Counter = 0;
int cnt = 0;
double totalHours = 0;
double totalCharges = 0;

//Global arrays
double arrayHours [100];
double arrayCharges [100];

//Functions
double calculateCharges ( double Hours );
int wHours ( int cnt );
int wCharges (int cnt );

int main() //begins function main
{
        cout << "Parking Charges Calculator."
                << endl; //informs that only whole hours are to be used

        while ( Hours >= 0 ) {  //gets Hours for each customer from user
                cout << "Enter parking time for this customer in hours (-1 to end): ";
                cin >> Hours;
                cout << endl;
                if ( Hours == -1 ) //terminated while before -1 added to array
                        continue;
                arrayHours [ Counter ] = Hours; // places Hours into arrayHours
                arrayCharges [ Counter ] = calculateCharges ( Hours ); //calls function to
calc charges
                Counter += 1; //adds 1 to counter
        } //ends while

        //tabulated output lines
        cout << "Car" << setw( 11 ) << "Hours" << setw( 10 ) << "Charge" << endl;
//header line
        for ( int cnt = 0; cnt < Counter; cnt++ ) //outputs all lines of arrays stored during
user input
                cout << cnt << setw( wHours ( cnt ) ) << fixed << setprecision (1)
                << arrayHours [ cnt ] << setw( wCharges ( cnt ) ) << fixed << setprecision
(2)
```

```
                << arrayCharges [ cnt ] << endl; //calls wHours and wCharges to set
column spacing and add to totals
        cout << "TOTAL" << setw( 9 ) << fixed << setprecision (1) << totalHours <<
setw ( 10 )
                << fixed << setprecision (2) << totalCharges << endl; //totals line

        return 0; //end function main
}

double calculateCharges ( double Hours ) //begin function calculateCharges.
//calcs the charges for current line and returns value
{
        double Charges = 0; //initialize Charges
        int Hrs = 0;
        Hrs = ceil ( Hours );
        if ( Hrs < 3 )  //if statement sets Charges to 2.00 if Hours less than 3
                Charges = 2.00;
        else //if Hours > 3, calculates charges, 2.  first 3 hrs and .50 for every 1 after
                Charges = ( ( Hrs - 3 ) * 0.50 ) + 2.00;

        if ( Charges >= 10.00 ) //limits Charges to 10.00
                Charges = 10.00;

        return Charges;

}  //end function calculateCharges

int wHours ( int cnt ) // begin function wHours.
//sets width of setw between car # and number of hours, adds current line value to total
{
        int   wH;
        wH= 13;
        totalHours += arrayHours [ cnt ];

        return wH;

}  //end function wHours

int wCharges ( int cnt ) //begin function wCharges.
//sets the width of the setw between Hours and Charges, adds current line value to total
{
        int wC;
        wC= 10;
        totalCharges += arrayCharges [ cnt ];

        return wC;
```

} //end function wCharges


"C:\Program Files\Microsoft Visual Studio\MyProjects\Parking\Debug\Parking.exe"

```
Parking Charges Calculator.
Enter parking time for this customer in hours (-1 to end): 4.5

Enter parking time for this customer in hours (-1 to end): 24

Enter parking time for this customer in hours (-1 to end): 24

Enter parking time for this customer in hours (-1 to end): 15

Enter parking time for this customer in hours (-1 to end): 6.9

Enter parking time for this customer in hours (-1 to end): -1

Car      Hours    Charge
0          4.5      3.00
1         24.0     10.00
2         24.0     10.00
3         15.0      8.00
4          6.9      4.00
TOTAL     74.4     35.00
Press any key to continue
```

Speculation, Conclusion, and Future Work

One question may arise as why online students use varieties of ways in solving problems which some are not common and understandable. One can conclude that online students use trial and error and their own logical skill to reach to the solution of the problem. There focus is to finish the job on given time. A limited access to instructor and classmate may contribute to finding independent solution that are totally unique and in some instances the solution look wrong but logically correct in contrast to in the face to face classes focus student who gather information and clue from the instructor and classmates and disseminated among each other. However, online variations in problem solving, is a subject worthy of further exploration which lead to draw a better conclusion.

References

Deitel P. (2005). *C++ How To Program, 5th Edition*, New Jersey: Prentice Hall.

Ebrahimi A. (2002) *C++ Programming Easy Ways*, Boston: American Press

Ury G. (2004) A Comparison Of Undergraduate Student Performance In Online and Traditional Courses. *Journal of Consortium for Computing Sciences in Colleges* 19, 4

Kleinman J., Entin E. (2002) Comparison Of In-Class and Distance- Learning Students Performance and Attitudes in Introductory Computer Science Course. *Journal of Consortium for Computing Sciences in Colleges* 17, 6