CSIS 112: Lab 1-Loan Payment Calculator

The purpose of this assignment is to get you back into programming and give you some practice with some C++ topics that you should be familiar with from CSIS 111 as well as a new concept, random number generation.

**Overview**:

You are working as a computer programmer for a mortgage company that provides loans to consumers for residential housing. Your task is to create an application to be used by the loan officers of the company when presenting loan options to its customers. The application will be a mortgage calculator that determines a monthly payment for loans.

The company offers 10-, 15-, and 30-year fixed loans. The interest rates offered to customers are based on the customer’s credit score. Credit scores are classified into the following categories:

**Table 1: Credit Score Categories**

**Rating** **Range**

Excellent 720-850

Good 690-719

Fair 630-689

Bad 300-629

The program should initially prompt the user (the loan officer) for the principle of the loan (i.e. the amount that is being borrowed). It should then ask him or her to enter the customer’s credit score. Based on the customer’s credit score, the program will randomly generate an interest rate based on the following ranges:

**Table 2: Interest Rate Assignments**

**Rating Interest Rate**

Excellent 2.75% - 4.00%

Good 4.01% - 6.50%

Fair 6.51% - 8.75%

Bad 8.76% - 10.50%

The final input should be the number of years that the loan will be outstanding. Because the company only offers three different terms ( 10-, 15-, and 30-year loans), the program should ensure that no other terms are entered.

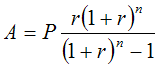
**Formulas**

**Payment Calculator**

[Adapted from Wittwer, J.W., "[Amortization Calculation](https://www.vertex42.com/ExcelArticles/amortization-calculation.html)," From [*Vertex42.com*](https://www.vertex42.com/), Nov 11, 2008.]

The formula to calculate the monthly payment for a fixed interest rate loan is as follows:

where



*A* = payment Amount per period

*P* = initial Principal (loan amount)

*r* = interest rate per period

*n* = total number of payments or periods

**Example:** What would the ***monthly*** payment be on a 15-year, $100,000 loan with a 4.50% ***annual*** interest rate?

**P** = $100,000   
**r** = 4.50% per year / 12 months = 0.375% (or 0.00375) per period   
**n** = 15 years \* 12 months = 180 total periods

A = 100,000 \* (.00375 \* (1 + .00375)180)/((1+.00375)180 – 1)

Using these numbers in the formula above yields a monthly payment of $764.99.

**Requirements:**

To receive credit for this assignment, certain programming features must be implemented in such a way as to demonstrate your knowledge of random number generation, use of enums, proper formatting of output, input error checking, switch statements, and if statements.

1. Start your program by prompting the user to enter a principle amount. The data type of this number should be a double. The amount must be positive and it must also be a numeric value. For example, when prompted to enter a number, if the user enters “abc,” the program will not be able to process the loan appropriately. Likewise, if the user enters -100000, the program will again produce erroneous results.
2. Prompt the user to enter the customer’s credit score. Again, appropriate error checking is essential here. In addition to ensuring that values are numeric and positive, you must also ensure that the credit score entered does not exceed 850. If a non-numeric, negative, or score that exceeds 850 is entered, prompt the user to re-enter the score.

If the credit score is below 300, display an error message to the user stating that the loan cannot be offered and exit the program. Make certain that the error message is displayed for a long enough duration for the user to read it before the program closes.

1. Prompt the user to enter the term of the loan. Valid terms are 10, 15, and 30 years. Any other numbers should be rejected, and the user should be prompted to re-enter an appropriate value.
2. Once the inputs have been entered and validated, your program must generate an interest rate to be used in the loan calculation.
   1. Create an enum to represent the possible credit ratings: EXCELLENT, GOOD, FAIR, and BAD.
   2. Assign the appropriate enum to the customer based on the ranges listed in Table 1 above. I recommend using if-statements for this.
   3. Using a SWITCH statement with the enum values as cases, assign an interest rate based on the customer’s credit rating.
      1. Within each case, use a random number generator to create an interest rate for each category based on the ranges listed in Table 2 above. Because the random number generator that you have been taught generates only integer values, you will need to generate a number in the hundreds and divide that number by 100 to get a result within the valid range for interest rates. Be careful with integer division here. Remember that an int divided by an int equals an int (i.e. int/int=int). To arrive at a double, you must make either the numerator or denominator a double to generate an interest rate with a decimal portion.
3. To perform the calculation for a monthly payment, create a function called CalcPayment that receives the principle, interest rate, and number of years as parameters. The function should return the monthly payment that is calculated.

Your program should check for invalid data such as non-numeric and non-positive entries for principle, credit score, and term. Use proper indentation and style, meaningful identifiers, and appropriate comments.

**General notes about error checking**

As in all applications, you should design your code to be robust enough to handle anything that a user may enter. In this program, you will prompt the user to enter three numbers: a principle amount, a credit score, and a number of years (i.e. term). Because users can make mistakes, you need to make sure that they enter numbers for these inputs (as opposed letters or other characters) and that any numbers entered are **reasonable**. For example, what if a user entered -10000 as the principle amount? Or negative years? What is a negative year anyway? It just doesn’t make sense.

Therefore, you may use the following code to determine if a non-numeric or negative number has been entered:

int num;

cout << "Enter an integer: " << endl;

cin >> num;

while (cin.fail() || num < 0)

{

cout << "You must enter a number, and that number must be positive. Please try again. " << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> num;

}

The reason that this code works is due to the behavior of “cin.” Whenever cin tries to read a letter (or any non-numeric character) into a variable that is designed to hold a number, cin enters a fail state, which can produce erroneous results and potentially even cause your program to crash. Therefore, you must trap for non-numeric data whenever the program is expecting a number. One way to trap for this error is to check if “cin” has entered the fail state using the “cin.fail()” function. If it has, then you first have to clear cin out of its fail state using the cin.clear() function. Afterwards, you can issue the “ignore” command which flushes any remaining garbage out of the input stream (aka “cin buffer”). Finally, you can prompt the user to re-enter a correct value. Note that the code above is not especially informative. That is, it produces a single generic error message to the user such that the user may not realize what he did wrong to cause the problem. Therefore, feel free to tweak it as necessary in order to customize your error messages. One other point of interest is that the code above doesn’t allow the user an opportunity to end the program if he can’t provide an acceptable value. It just continues to loop until an acceptable value is entered. In real life, you would definitely want to allow your user a means of escape. In this assignment, however, continuing to prompt the user for a valid value until one is entered is fine. Later in the course, we’ll learn more sophisticated ways of error checking when we discuss exception handling.

Good luck on this assignment! Have fun with it, and as always, let me know if you have any questions.

**Deliverables**:

* Complete the programming assignment described above and submit your completed assignment in accordance with the assignment submission policies.