# Description

The purpose of this lab is to check your understanding of the content in lecture 2a, Algorithms and Control Statements.

We will focus on the material from chapter 4 of the book. The objectives are:

* Basic problem-solving techniques.
* To develop algorithms through the process of top-down, stepwise refinement.
* To use the if and if…else selection statements to choose among alternative actions.
* To use the while repetition statement to execute statements in a program repeatedly.
* Counter-controlled repetition and sentinel-controlled repetition.
* To use the increment, decrement, and assignment operators.

# Part 1: Questions

Complete the assignment in Topic 2 named, "Lab 2a Questions", in Canvas.

# Part 2: Activity

This part of the lab should be submitted using the Lab 2a Activity assignment in the "Topic 2a Lab: Algorithm Development and Flow Control" module.

## Section 1: Correct the code

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, circle the error in the program and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [*Note:* It is possible that a program segment may contain multiple errors.]

1. Rewrite this program to give the proper output. Write the complete corrected version.

if (grade >= 60)

cout << "Passed.\n"

else

cout << "Failed.\n"

cout << "You must take this course again.\n"

Answer a: There is one compilation error, involving braces on the else statement. There is also a syntactic error because there are no semicolons after the cout statements.

if (grade >= 60)

cout << “Passed. \n”;

else

{

cout << “Failed. \n”l

cout << “You must take this course again. \n”;

}

1. The following program segment should input 15 integers from the user and calculate their total. Write the complete corrected version of the code.

int total = 0;

int counter = 1;

int input;

while (total <= 15)

{

cin >> input;

total += input;

counter++;

}

Answer b: There is a logical error inside of the conditions for the while loop, instead of the total we should be using the counter, as that is the count of inputs.

int total{0};

int counter{1};

int input{0};

while (counter <= 15)

{

cin >> input;

total += input;

counter++;

}

1. The following program segment should input and sum integers from the user until the sentinel value, -1, is entered. Write the complete corrected version of the code.

int total = 0;

int input;

while (input != -1)

{

cin >> input;

total += input;

}

Answer c: In this case, we should prompt the user for an input. Input is also uninitialized, so there may be a case where the loop is never initialized. A do while loop may be the best choice for this code.

int total{0};

int input{0};

do

{

cout << “Please enter integer values (-1 to exit)”;

cin >> input;

total += input;

}while (input != -1);

1. The following program segment should output the number 9. Write the complete corrected version. Write the complete corrected version of the code.

int x = 4;

cout << (x++ + 4);

Answer d: In this case the original code has the ‘x’ post-incremented, meaning the increment happens after the operation. We should instead use the pre-increment to increment ‘x’ before its operation.

int x = 4;

cout << (++x + 4);

1. The following while loop should compute the product of all integers between 1 and 5, inclusive. Write the complete corrected version of the code.

int i = 1;

int product = 1;

while (i <= 5);

product \*= i;

Answer e: Rather than using a while loop, a for loop should be used to iterate the loop 5 times.

int i = 1;

int product = 1;

for ( i; i <= 5; i++)

{

product \*= i;

}

## Section 2

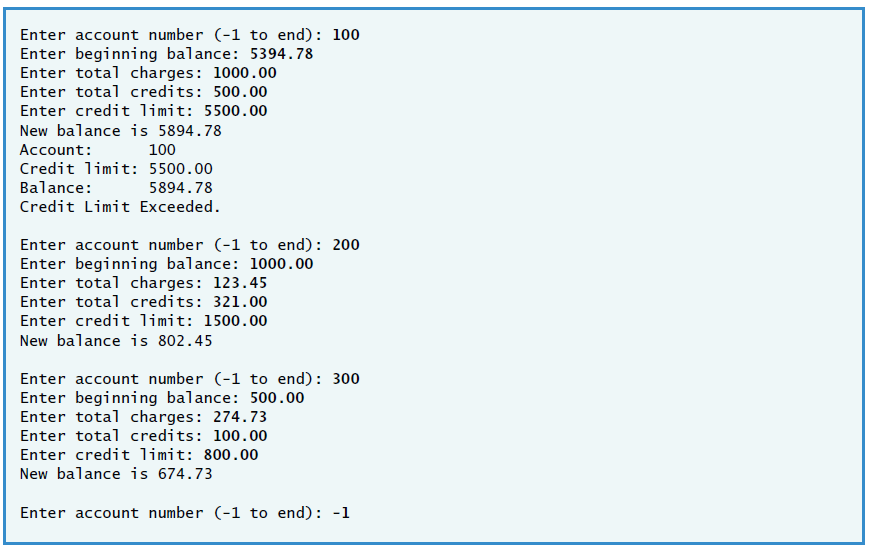
Write a C++ program that will determine whether a department-store customer has exceeded the credit limit on a charge account. For each customer, the following information is available:

* Account number (an integer)
* Balance at the beginning of the month
* Total of all items charged by the customer this month
* Total of all credits applied to the customer’s account this month
* Allowed credit limit.

The program should **use a while loop** to input each of these facts, calculate the new balance (= beginning balance + charges – credits) and determine whether the new balance exceeds the customer’s credit limit. For those customers whose credit limit is exceeded, the program should display the customer's account number, credit limit, new balance and the message "Credit Limit Exceeded." **The source code file should be called, "credit\_limit.cpp".**

**Your output should be formatted as seen below, do not improvise on this formatting.**

A sample run of the program. Note: this sample contains the prompts as well as the user's input:



### Tips

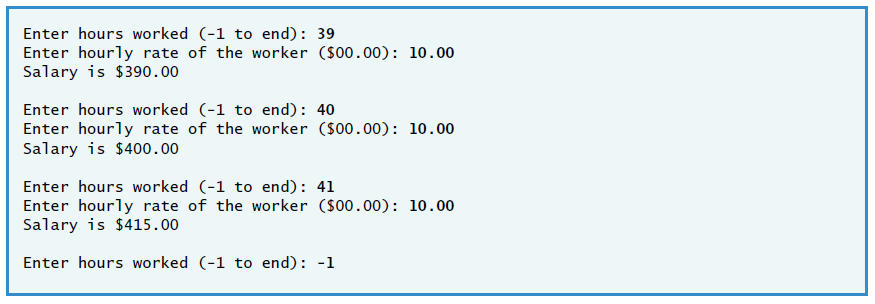
* Notice that the number of customers is not specified in advance. The problem statement implies that you should use a sentinel-controlled the loop.
* The input data consists of one integer and four monetary amounts. The monetary amounts are numbers with decimal points, so you will use type double to represent them.
* You will use a while loop to process several sets of customer data.
* What sentinel value should you use? It needs to be a value that will not be confused with a legitimate data value. A good choice for this program would be to use -1 for the customer’s account number, because account numbers are nonnegative integers.
* Before the loop, you will prompt for the account. Inside the loop, you will prompt for the monetary amounts. It is a good practice to remind the user of the sentinel value in each prompting message. After processing the data, you obtain the next account number.
* While you could prompt each time for all five facts needed for a customer, a better strategy would be to prompt just for the account number. If it is negative, terminate the loop. If it is nonnegative, prompt for the remaining four pieces of data for that customer.
* How do you determine whether the new balance exceeds the credit limit? You have already input the credit limit, but you will need to calculate the new balance which is equal to the starting balance, plus the charges, minus the credits.
* Be sure to follow the spacing and indentation conventions mentioned during lecture. Before and after each control statement, place a line of vertical space to make the control statement stand out. Indent all the body statements of main and indent all the body statements of each control statement.
* Print your outputs neatly in the indicated format. Use setprecision( 2 ) to force two positions of precision to the right of the decimal point when printing monetary amounts. You will need to #include <iomanip> to use function setprecision( 2 ).
* Also remember to use the fixed manipulator. This manipulator specifies that the number output should be output in fixed notation, as opposed to scientific notation (i.e., scientific). Fixed notation always keeps a “fixed” decimal point, regardless of the number of digits displayed. For example, 1.50000000000 is expressed in fixed notation. Scientific notation displays numbers in a format such as 1.5E+10.
* Be sure to use the template file associated with this lab.

## Section 3

Write a C++ program that **uses a while loop** to determine the gross pay for each of several employees. The company pays “straight time” for the first 40 hours worked by each employee and pays “time-and-a-half” for all hours worked in excess of 40 hours. You are given a list of the employees of the company, the number of hours each employee worked last week and the hourly rate of each employee. Your program should input this information for each employee and should determine and display the employee's gross pay. **Your source code file for this program should be called, "gross\_pay.cpp".**

**Your output should be formatted as seen below, do not improvise on this formatting.**

A sample run of the program. Note: this sample contains the prompts as well as the user's input:



### Tips

* Notice that the number of employees is not specified in advance. The problem statement implies that you should use a sentinel value to control the loop.
* The input data consists of two doubles per employee, one for the number of hours worked and one for the hourly rate.
* You will use a while loop to process several sets of employee data.
* Before the loop, you will prompt for the number of hours worked. Inside the loop, you will prompt for the hourly rate. It is a good programming practice to remind the user of the sentinel value in each prompting message. After processing the data, you obtain the next employee’s number of hours worked.
* What sentinel value should you use? It needs to be a value that will not be confused with one of the legitimate data input values. A good choice for this program would be to use –1 for the employee’s number of hours worked, because the number of hours worked must be a nonnegative value.
* While you could prompt each time for both pieces of information needed for a customer, a better strategy would be to prompt just for the number of hours worked. If it is -1, terminate the loop. If it is not -1, prompt for the remaining piece of data for that employee, namely, the hourly rate.
* How do you calculate the pay for each employee? If the person worked 40 hours or less, multiply the number of hours worked times the hourly salary; otherwise (else), calculate the pay by multiplying 40 times the hourly salary rate and adding to this value to the number of overtime hours times the overtime rate. The number of overtime hours is the number of hours in excess of the first 40 hours worked (i.e., hours – 40), and the overtime salary is 1.5 times the hourly rate.
* Be sure to follow the spacing and indentation conventions mentioned in the text. Before and after each control statement, place a line of vertical space to make the control statement stand out. Indent all the body statements of main and indent all of the body statements of each control statement.
* Print your outputs neatly in the indicated format. Remember to use setprecision( 2 ) to force two positions of precision to the right of the decimal point when printing monetary amounts. You will need to #include <iomanip> to use function setprecision( 2 ).
* Also remember to use manipulator fixed. This manipulator specifies that the number output should be output in fixed notation, as opposed to scientific notation (scientific). Fixed notation always keeps a “fixed” decimal point, regardless of the number of digits displayed. For example, 1.50000000000 is expressed in fixed notation. Scientific notation displays numbers in a format such as 1.5E+10.
* If you have any questions as you proceed, ask your lab instructor for help.

# What to Submit for Part 2-Activity

* This document with all the problems solved from section 1.
  + Your answers must be in the boxes provided.
  + Do not submit hand-written work.
* Screenshots of VSCode with the output of both programs visible.
* The program that you created for section 2 called, "credit\_limit.cpp".
* The program that you created for section 3 called, "gross\_pay.cpp".