# Description

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

We will focus on the material from chapter 5 of the book. The objectives are:

* The essentials of counter-controlled repetition.
* To use the *for* and *do-while* repetition statements to execute statements in a program repeatedly.
* To implement multiple selection using the switch selection statement.
* To use the break and continue program control statements to alter the flow of control.
* To use the logical operators to form complex conditional expressions in control statements.
* To avoid the consequences of confusing the equality and assignment operators.

# Part 1: Questions

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

# Part 2: Activity

This part of the lab should be submitted using the Lab 2b Activity assignment in the "Topic 2b Lab: Algorithm Development and Flow Control" module. It is possible that some of the images and answer-boxes below might move while using them. Do your best to prevent this from happening or fix it when it does.

## Problem i: Code Output

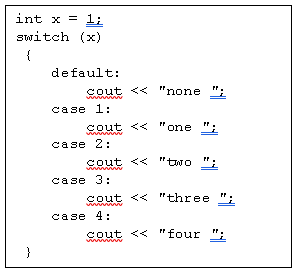
For each of the given program segments, read the code and write the output in the box provided below each program. [*Note:* Do not execute these programs on a computer.]

1. What is output by the following *switch* statement?

The output should be:

None one two three four

Because there are no breaks after each case the statement falls through each case, outputting each case.

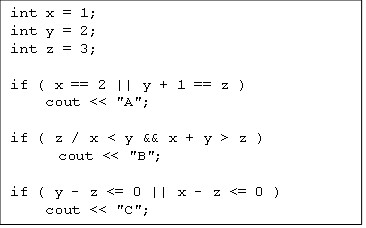


1. What is output by the following for loop?

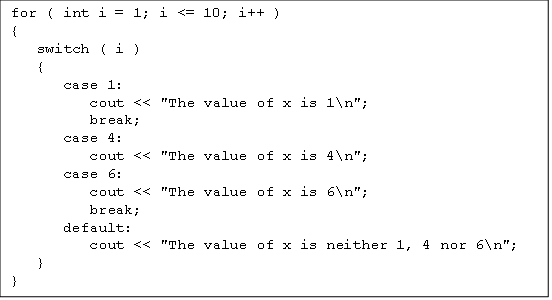
problem 2

0 1 2 3 4

1. What is output by the following program segment?



AC

1. What is output by the following program segment?

The value of x is 1

The value of x is neither 1, 4 nor 6

The value of x is neither 1, 4 nor 6

The value of x is 4

The value of x is 6

The value of x is neither 1, 4 nor 6

The value of x is 6

The value of x is neither 1, 4 nor 6

The value of x is neither 1, 4 nor 6

The value of x is neither 1, 4 nor 6

The value of x is neither 1, 4 nor 6

1. What is output by the following program segment? problem 5

1 2 3 4 5 6 7

The final value of x is: 7

1. problem 6What is the output of the following program segment?

The value of x is: 1

The value of x is: 2

The value of x is: 3

The value of x is: 4

The value of x is: 5

The value of x is: 6

The value of x is: 7

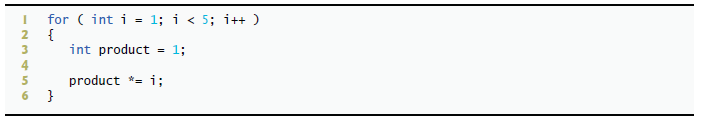
The value of x is: 8

The value of x is: 9

The value of x is: 10

Problem ii: Correct the Code

For each of the given program segments, determine if there is an error in the code. If there is, 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. The following program segment should calculate the product of the integers between 1 and 5, inclusive.

If the intent is to calculate the product of integers between 1 and 5 inclusively, then on line 1 the test for the for loop should be “i <= 5”

Also, because the product is being declared within the loop causes it to be reset every time. The solution is to declare and calculate the product outside of the loop.

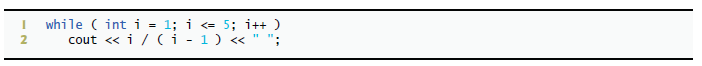
int product = 1;

for (int i = 1; i <= 5; i++)

{

product \*= i;

}

1. The following for loop should divide i by i - 1, using integer division, and print the result.

A for loop should be used instead of a while loop, since while loops operate on Boolean logic there should only be things that evaluate to true/false within the parameters of a while loop. Also the first iteration of this loop will divide by zero.

for (int i = 1; i <= 5; i++)

cout << i / ( i – 1 ) << “ “;

1. The following for loop should print all the integers between 5 and 1000, inclusive, that are evenly divisible by 5.

This is a logic error.

for( int i = 5 ; i <= 1000 ; i++ )

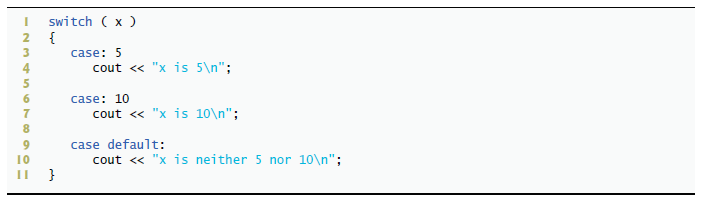
{

if (i % 5 == 0)

cout << i << “ “;

}



1. The following switch statement should print either x is 5, x is 10 or x is neither 5 nor 10.

This is a logic error, there are no breaks between each case so the output will fall through each case. Also the case number should come before the colon.

switch (x)

{

case 5:

cout << “x is 5\n”;

break;

case 10:

cout << “x is 10\n”;

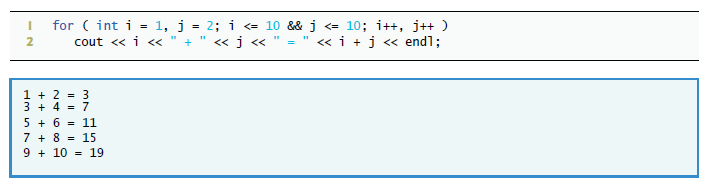
break;

default :

cout << “x is neither 5 nor 10\n”;

break;

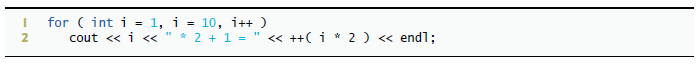
}

1. The following program segment should print the sum of consecutive odd and even integers between 1 and 10, inclusive. The expected output is shown below the code segment.

for (int i = 1, j = 2 ; i <= 10 && j <= 10; i++, j++)

if ( i % 2 != 0 && j % 2 == 0 )

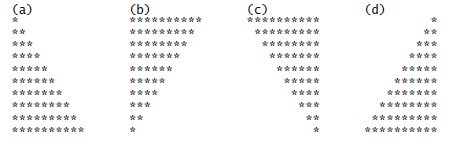
cout << i << “ + “ << j << “ = “ << i + j << endl;

1. The following for loop should compute the product of i times 2, plus 1. For example, if the counter is 4, the program should print 4 \* 2 + 1 = 9. It should loop from 1 to 10.

for ( int i = 1; i <= 10; ++i )

cout << i << “ \* 2 + 1 = “ << ( i \* 2 ) + 1 << endl;

### Problem iii: Write the Code

1. Write a program **called, "patterns.cpp",** that uses for statements to print the following patterns separately, one below the other. Use for loops to generate the patterns. All asterisks (\*) should be printed by a single statement of the form cout << '\*'; (this causes the asterisks to print side by side). [*Hint:* The last two patterns require that each line begin with an appropriate number of blanks.] Use the template file for Labs to create this program.  **Only include iostream.**
2. A right triangle can have sides that are all integers. A set of three integer values for the sides of a right triangle is called a **Pythagorean triple**. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse. Write a program **called, "py\_tripple.cpp",** to find all Pythagorean triples for side1, side2 and side3 (the hypotenuse) all of which are no larger than 500. **The three sides should satisfy the following inequality, side1 <= side2 < side3 <= 500.** This will prevent repeated (mirrored, or rotated) triangles.

**Try to make your code as efficient as possible. Avoid wasted loop iterations.**

**Hint:** you could use three nested loops to solve this problem but might be able to accomplish the task with two. Regardless of your approach, be sure to satisfy the relationship between the sides.

**Note:** This is an example of “brute force computing.” You will learn in more advanced computer-science courses that there are many interesting problems for which there is no known algorithmic approach other than using sheer brute force.



# What to Submit for Lab 2b Activity

* This document completed.
* The source code file called, "patterns.cpp".
* The source code file called, "py\_triplets.cpp".
* A screenshot of the output for *patterns.cpp*.
* A screenshot of the output for *py\_tripplets.cpp*.

Note: your screenshots must include the entire Visual Studio Code window. Do not include your desktop or anything else in the image. Do not take a picture with a camera or phone, use your computer to create a screenshot.