Fundamentals: Drills

Writing and running a new program

1)
    a. Write a program that outputs “Hello, world!” to the screen.
    b. Remove the #include <iostream> from the above program. What do you expect will happen? What happened?
    c. Replace the #include <iostream> line but this time delete the “using namespace std” line. What happens?
    d. Deeper challenge: instead of restoring that line, add the string std:: to your cout directives. They should look like: std::cout << … Does that work? If you get any further errors, for example with the endl symbol, see if adding std:: will fix that, too.
2) Print the names “Alice”, “Bob”, and “Clarice” to the screen, one per line, in as many different ways as you can think of.

Declaring variables

1)
    a. What happens when you try to name a variable with a name that **starts with** a special character or a number?
    b. What happens if you try to name a variable using a keyword? For example, what happens if you try to compile the line:

        int int = 4
2)
    a. Declare three doubles that relate to being a student with useful variable names on a single line, and initialize them all to different values.
    b. Declare two doubles, three ints, and two characters in as many different ways as you can think of. For example, you may consider declaring one per line, or multiple variables per line. Which method do you prefer?
3) Declare an integer and do not initialize it. Output it to the screen. Compare your results to a many people as possible. Do the all of the results agree? What does this mean?

Working with variables: changing their values with math

1)
    a. Declare three integers. Read in their values using only a single cin statement.
    b. Repeat the drill above using three separate cin statements. Which method do you prefer? (Hint: this is a personal style choice, not a hard and fast rule!)
2) Declare a constant double that converts radians to degrees, and another that converts degrees to radians. Have the user enter a value in radians. Convert the value to degrees and store the

result. Output the result, then convert it back to radians and output the result. Does the last output always agree with the input? Test it until you are satisfied with your answer.

3) In this drill you are going to compute the value of a polynomial using a program. Determine the value of y in the following expression:

$$y = ax^2 + bx + c$$

The user must enter values for the coefficients a, b, and c, and then a value for the variable x. Confirm that your code is correct by performing the same calculation by hand.

4) In this drill you are going to compute the value of a rational expression using a program. The user must enter a value for x, and your program must output the value for y. Confirm that your code is correct by performing the same calculation by hand. The expression is:

$$y = \frac{2}{3}\,\frac{x+4}{x-10}$$

Did your output agree with what you computed by hand? What happens if you enter x =10?

5) What is the value of the variable "sum" output by the following code snippet? Why?

```
int a = 0;
int b = 0;
int sum = a + b;
cout << "Enter values for a and b: "<<endl;
cin >> a;
cin >> b;
cout << "The sum is: " << sum;
```

The unique problems and opportunities with integers and mathematics

1)

a. Have the user enter two integers. Output their quotient to the screen. Come up with test cases where:
   i. The result is correct according to the rules of division that you know
   ii. The result is incorrect due to integer division errors

b. Have the user enter two integers. Output their actual quotient to the screen in as many ways as you can think of. Hint: you'll have to convert something to a double or a float. How many ways are there to do that?

2) The following code was submitted by a student who claimed that integer division was avoided. The student is incorrect. What went wrong in his reasoning? Determine two test cases that can be used to prove that integer division is still encountered. Fix the code so that integer division is indeed avoided.

```
#include <iostream>

using namespace std;

int main()
{
    int a, b;
```

```
    double result;
    cout << "Enter values for a and b!" << endl;
    cin >> a >> b;
    result = a/b;
    cout << result;
}
```

3)

a. Have the user enter an integer and print the last digit of that integer to the screen. There are at least two ways to do this. Solve this problem both ways.
b. Have the user enter an integer that has exactly three non-zero digits. Output each digit to the screen, one per line.
c. Have the user enter an integer that has exactly three non-zero digits. Reverse the digits and store the results in an integer. Add it to the original integer and print the results to the screen. For example, if I enter 567, the reversed integer is 765, and their sum would be 1332, which gets output to the screen.

4) In this drill you are going to explore the behaviour of the modulo operator. Write a program that allows the user to enter two integers, a and b. Output a%b to the screen. Experiment with different values of a and b until you can begin to answer the following questions:

a. If a < 0 but b > 0, is a%b positive or negative?
b. If a < 0 and b < 0, is a%b positive or negative?
c. If a > 0 but b < 0, is a%b positive or negative?

The C++ math library

1) We know from trigonometry that the equation below should always be true:

$$\sin(x)^2 + \cos(x)^2 - 1 = 0$$

However, the computer can only approximate the values of sin and cos. Write a program that prompts the user to enter a value for x and evaluate the above expression. Test it with as many test cases as you can, and output the result each time. Does it always equal to zero? (Note: nowadays, computers are getting better and better at this approximation, so it may be difficult to find a test case for which the equation doesn't hold. That's not a bad thing!)\

2) Predict the output of the following code, then test the code and see if your output was correct. If not, what went wrong?

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

using namespace std;

int main()
{
    cout << cos(0) << endl;
    cout << sin(M_PI/2) << endl;
    cout << cos(90) << endl;
    cout << 1/2*sin(M_PI/2) << endl;
    cout << pow(10,1/2) << endl;
    cout << sqrt(225) << endl;
    cout << sqrt(-1) << endl;
```

```
}
```