CS 31 Worksheet Week 2

This worksheet is entirely **optional**, and meant to prepare you for upcoming projects and exams. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. We encourage you to collaborate with your peers while completing this worksheet. If you finish the worksheet early, you are welcome to inquire with your LA for this week's Supplemental Problems.

Note: All exams will be completed on paper, we highly encourage you to complete these problems on paper

Topics Covered: If Statements, Cin, Variables, Doubles, Ints

Solutions are written in red. The solutions for **programming** problems are not absolute. There are many equally correct solutions and different valid perspectives when it comes to solving Computer Science problems. If your solution is different, we highly recommend you share that with students as well!

Reading Problems

1. Assume that the following lines of code are inside the main function, with #include <iostream> and using namespace std, and all the string variables used have been previously declared.

(a) Highlight or box where the bug occurs.
(b) Explain what you think will happen when running the program.
(c) Is this a logic error or a compilation error? Why?
(d) Add a fix to the problem you found in part (a).

```
string name;
cout << "Enter your name: ";
getline(cin, name);

int UID;
string major;

cout << "\nEnter your UID: ";
cin >> UID ;
```

```
cout << "\nEnter your Major: ";
getline(cin, major);

string hall;
cout << "\nEnter your residence hall: ";
getline(cin, hall);

cout << "\n" << UID << " is the ID of " << name
     <<", a " << major << " student who lives in " << hall << endl;
```

Time: 10 min

a) Bug: Program will skip “Enter your Major”, because getline has already consumed a newline character.

b) A newline is always appended to your input when you select Enter or Return when submitting from a terminal. It is also used in files for moving toward the next line. When the flow of control reaches std::getline(), the newline will be discarded, but the input will cease immediately. The reason this happens is because the default functionality of this function dictates that it should (it attempts to read a line and stops when it finds a newline).

c) This is more of a logic error than a compilation error because it will not stop your program from compiling. However, because we have made a mistake in predicting what the computer will do, we have made an error in our logic. Errors like this can sneak up on you without the compiler letting you know.

d) Because this leading newline inhibits the expected functionality of your program, it follows that it must be skipped or ignored somehow. One option is to call cin.ignore(10000, ‘\n’) after the first extraction. It will discard characters up to the next newline so that the newline is no longer intrusive – the 10,000 in cin.ignore just means that you include up to 10,000 characters before the newline; meant to represent an arbitrarily long line.

2. What is the output of the following code?

```
int cookies = 12;
int mms = 120;
if (mms % cookies != 0) {
    cout << "Can’t evenly split M&Ms for each cookie!" << endl;
} else {
    cout << "We have " << mms/cookies << " M&Ms per cookie." << endl;
}
```

Time: 3 min

Solution: “We have 10 M&Ms per cookie.”

This is because **the modulo operator ‘%’ returns the remainder** of the two variables. Ex: 5 % 2 is the remainder of 5/2. In this case, 120/12 = 10 with no remainder. Thus, 120%12 = 0, which does not pass the if statement, and continues to the else and prints out that line. If the remainder was not 0, then it would run only the if statement. Outputs the resulting string, with the quotient inserted.

3. This code snippet takes a certain “hour” and “weekday” and tries to tell you if you can buy turnips from Daisy Mae, the turnip seller.

(a) Find the 7 lines with mistakes in the code and fix them.
(b) Will this code compile? Why or why not?
(c) After you fix the bugs, imagine you input 10 for the hour, then “Sunday” for the weekday. What will this program say to you?

```
int hour;
string weekday;
cin << hour;
cin << weekday;

if (weekday != "Sunday" || hour >= 12)
   cout >> "Daisy Mae is not here!" >> endl
else {
   if (hour = 11) {
       cout >> "It's almost 12! Hurry up!" >> endl;
   } else {
       cout >> "Buy turnips with Bells." >> endl;
}
```

Time: 10 min

Solution:

(a)

```
int hour;
string weekday;
cin >> hour;
cin >> weekday;

if (weekday != "Sunday" || hour >= 12)
   cout << "Daisy Mae is not here!" << endl;
```

```
else {
   if (hour == 11) {
       cout << "It's almost 12! Hurry up!" << endl;
   } else {
       cout << "Buy turnips with Bells." << endl;
   }
}
```

(b)

This code will not compile!

Note: The line with the conditional (hour = 11) will NOT be a compile error but would be a logic error.

Rather than (hour == 11), which checks whether **hour** is equal to **11**, hour is SET to 11 via (hour = 11) and then is evaluated for its truthiness. **if (hour = 11)** is an equivalent expression to **if (11)**. This always evaluates to true, and is thus a LOGIC ERROR.

All of the other errors are COMPILATION ERRORS. It's important to be very careful as you read and write code on paper. There won't be any compiler to warn you during the exam that you switched the symbols for cout and cin, or that you forgot one bracket or equals symbol, or that you forgot a semicolon! :(

(c)

Buy turnips with Bells.

4.

(a) Highlight where the bug occurs.
(b) Explain what you think will happen when running the program and why.
(c) After deleting the bugged line of code, what will the remaining program output? Can you explain every line of output?

```
#include <iostream>
using namespace std;

int main() {
    int elligent = 64;
    int eresting = 0;
    double rainbow = 64.0;
    double stuf = 0.0;

    cout << elligent << endl;
```

```
    cout << rainbow << endl;

    eresting = elligent/2.5;
    stuf = rainbow/3;
    cout << rainbow/3 << endl;
    cout << stuf << endl;
    cout << elligent/2.5 << endl;
    cout << elligent/(rainbow-64) << endl;
    cout << eresting << endl;
}
```

Time: 7 min

Solution:

(a)
Second last line where we cout elligent/(rainbow-64), which is a double value divided by (64.0 - 64 = 0.0)

(b)
The code will run normally until it hits the second last line, where it attempts to divide by 0, which is undefined behavior. The likeliest thing that would happen is that the program crashes.

(This is also known as a runtime error or floating point exception, though that may be out of the scope of the class right now)

(c)

```
64
64
21.3333
21.3333
25.6
25
```

64 (Printing an integer is simple)
64 (Even if we print a double, because 64 is a whole number, it just prints as 64)
21.3333 (This is a double decimal value.)
21.3333 (This is also a double decimal value.)
25.6 (Even though elligent is an integer, if we use it in an operation, the result will be a decimal/floating point value!)
25 (This case is different because we are trying to assign a value to an int variable. Even if the previous result was 25.6, and we're

doing the same operation, when we put that value into an integer variable, the “decimal point” part will get truncated, leaving the whole number 25.)

When we declare a variable as a certain type, the computer knows to treat that variable as that type.

Programming Problems

1. Write a program that asks for a number between 0 and 100 (exclusive), and takes an integer input. If you input a number greater than or equal to 100, it will print “Liar, liar, plants for hire”. If you input a number less than or equal to 0, it will print “Liar, liar, plants for hire”

After this, if your number is at least 50, and will print “Almost to 100!” (in addition to the above possibilities)Otherwise, it will print “Still a-ways to go!”

Example Output:
Case one:
Please give me a number between 0 and 100
0
Liar, liar, plants for hire.

Case two:
Please give me a number between 0 and 100
95
Almost to 100!
Case three:
Please give me a number between 0 and 100
45
Still a-ways to go!
Case four:
Please give me a number between 0 and 100
105
Liar, liar, plants for hire.
Almost to 100!

Time: 10 min

Solution:

```
int num = 0;
cout << “Please give me a number between 0 and 100” << endl;
cin >> num;

if(num >= 100 || num <= 0) {
   cout <<  “Liar, liar, plants for hire” << endl;
}

if (num >= 50) {
   cout << “Almost to 100!” << endl;
} else {
```

```
    cout << "Still a-ways to go!" << endl;
}
```

2. Write a program that takes in two numbers and a command of type string ("Add", "Subtract", "Multiply", "Divide"). Inputting an invalid command should cause the program to print out “Invalid command!” and stop.

Sample output:
Enter your first number: 3
Enter your second number: 7
Enter your command: Multiply
Result: 21

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    int first = 0;
    int second = 0;
    string command = "";

    cout << "Enter your first number: ";
    cin >> first;

    cout << "Enter your second number: ";
    cin >> second;
    // ignore the next \n to avoid complications!
    cin.ignore(1000000, '\n');

    cout << "Enter your command: ";
    getline(cin, command);

    // Check the string to each possible type of command
    if (command == "Add")
        cout << "Result: " << first + second << endl;
    else if (command == "Subtract")
        cout << "Result: " << first - second << endl;
    else if (command == "Multiply")
        cout << "Result: " << first * second << endl;
    else if (command == "Divide" && second != 0)
        cout << "Result: " << first / second << endl;
    else {
        cout << "Invalid command!" << endl;
    }
}
```

3. (This may be out of scope for this week, but you can try it as a challenge question!) Write a program that takes in a number as an int and outputs the sum of all of the digits in that number.

Sample Output:
Enter a number: 184
The sum of the digits in your number is 13!

```
#include <iostream>
using namespace std;

int main() {
	cout << "Enter a number: ";

	int num = 0;
	cin >> num;

	int sum = 0;

	/* We want to keep adding until we run out of digits, so we can
	use a while loop instead of a for loop here */
	while (num > 0) {
		sum += num % 10;
		num /= 10; //or num=num/10;
	}

	cout << "The sum of the digits in your number is " << sum <<
	"!" << endl;
}
```

Conceptual Problems

1) When using cin, the standard notation is

```
int n;
cin >> n;
```

However, when using cout, the standard notation is

```
int n = 5;
cout << n;
```

Why do you think the angle brackets are pointed right for *cin* but left for *cout*? How might this help you remember the correct direction?

Solution:

The library designer decided to use angle brackets to visually suggest the direction that data is moving. In cin >> n, that would be from cin, the standard input source, to n. In cout << n, the direction would be from n to cout, the standard output destination.

Simply put, the angle brackets serve as redirection operators to send data from one place/variable to another. So data is redirected from cin, the standard input stream where the user enters data, to variable n, hence the right pointing brackets. Similarly, data is redirected from the variable into *cout* (standard output stream) so that it can be printed, hence the left pointing brackets.