In-class Worksheet #2 and HW2
Fall 2022
Computer Science I

==========
Programming:

1. Analyzing problem/requirement
2. Breaking it down to steps
3. Express your idea (step-by-step approach to solve the problem) in C++ code
   ==> Require your understanding of C++ language,
   ==> e.g., how to declare a variable? How to do computations… How to branch? How to repeat? …
4. Compile your code and fix any compilation errors (or syntax) errors.
5. Run your executable, and try different inputs and check if the program generates the expected (correct) output
   Chances are there are mistakes (bugs) in your code, and you need to fix them (debug).

==========

Bug was used when a moth caused a failed relay on the Harvard Mark 1 computer. Grace Hopper and other programmers taped the moth in logbook stating: "First actual case of a bug being found."

Different Errors: Syntax errors, run-time errors and logic errors

- **Syntax errors**
  - Violation of the **grammar rules** of the language
  - Reported by the compiler (i.e., shown when you use g++ to compile your .cpp file)
  - Error messages may not always show correct location of errors

What are the grammar rules we have learnt (implicitly or explicitly)? Pay attention to C++ syntax rules when writing your program…

* Structure of the program… so far just memorize it!

* each statement ends with ;

* Usage of quotation marks,

* a variable must be declared first, then it can be used

* syntax of cin statement, cout statement?

* syntax of expressions: including common sense such as matching parenthesis ( )

* syntax of if statement, if/else statement, switch statement, while loop, do/while loop

* syntax of compound statement

- **Run-time errors**
  - Error conditions detected when the executable is run, e.g., when a number is divided by a variable that has a value of 0; or trying to access an array with an index that is out of boundary. The computer will reports such errors.
- **Logic errors**
  - Errors in the program's idea, or some mistaken usage of C++ language (leading to your code doing something other than you have in mind)
  - Most difficult to diagnose
  - Computer does not recognize such errors

Whitespace independent language and Coding Style

Whitespace is a term that refers to characters that are used for formatting purposes. In C++, this refers primarily to spaces, tabs, and newlines.

Whitespace is required to separate adjacent words and numbers; they are ignored everywhere else except within quotes and preprocessor directives. For this reason, we say that C++ is a whitespace-independent language.

Consequently, the following statements all do the exact same thing:

```
std::cout << "Hello world!";

std::cout           <<          "Hello world!";

                std::cout <<          "Hello world!";

std::cout
        << "Hello world!";
```

As programmer, you should pay attention to the usage of whitespace (indentation and blank lines) as they help to make the code readable, which in turns helps with debugging. Often this is the first step to fix tricky logic errors!

Indentation: add whitespace to lines within a { }. Emacs takes care of this for you!
* For example, statements in main( ) are indented by a certain # of spaces or tab.
* Indent by one more level the statements to be chosen in if/else, if statement

Practice Problems.

1. Find at least five syntax errors (i.e., compilation errors) in the following programs.

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

int main();
{
  cout << "Please enter two numbers:"
  cin << x, y;
  cout << "The sum of << x << "and" << y
      << "is: " x+y << "\n";
  return;
}
```

2. Find at least three logic errors in the following program that read in two integers and calculate the average of the two numbers (keeping the decimal parts), i.e., if the two numbers entered are 1 and 2, the program should calculate the average to be 1.5.

```
#include <iostream>
using namespace std;
int main() {

    int total;
    int x1;
    cout << "Please enter a number:";
    cin >> x1;
    total = total + x1;
    cout << "Please enter enother number:";
    int x2;
    cin >> x2;
    total = total + x1;
    float average = total/2;
    cout << "The average of the two numbers is "
         << average << "\n";
    return 0;
}
```

3. Find logic errors in the example code below. Hint: first fix the indentation!

a)

```
if (pizza12<0);
 {
     cout<<“Cannot order negative num of pizza!\n”;
     return 1;
 }
```

b) if x is 1, increase y by 1; if x is 2, increase y by 2

```
if (x=1) y++; else if (x=2) y=y+2;
```

C) test if x does fall inside the range of [1,10] …

```
if (1 <=x <=10) cout <<”x is in the range of [1,10]\n”;
```

d). Check if string variable s stores a valid US coin name

```
if (s != “nickel” || s!=”penny” || s!=”dime” || s!=”quarter”)
    cout <<”Wrong name for coins\n”;
```

e). by default language is English. If the country is USA and the state is PR, then set the language to Spanish. If the country is Japan, set the language to Japanese.

```
string language=”English”;
if (country==”USA”)
    if (state==”PR”) language=”Spanish”;
    else if (country==”Japan”)
language=”Japanese”;
```

5. What will be the output of the following code segment, if the value of variable *richter* is 7.2? How can you fix the program? Can you fix the program by just modifying the conditions, while keeping the cout statements in their current places?

The program is supposed to interpret the value of richter by printing out the kind of damage associated with different values:

If ritcher is between 4.5 and 6, then display “Damage to poorly constructed buildings”;

if ritcher is between 6.0 and 7.0, then display “Many buildings considerably damaged, some collapse”;

if ritcher is between 7.0 and 8.0, display “Many buildings destroyed”; and

if ritcher is larger than 8.0, display “Most structures fall”.

```
if (richter >= 4.5)
cout <<”Damage to pooly constructed buildings”;
else if (richter >= 6.0)
cout <<”Many buildings considerably damaged, some collapse”;
else if (richter >= 7.0)
cout <<”Many buildings destroyed”;
else if (richter >= 8.0)
cout <<”Most structures fall”;
```

6. The following piece of code does not follow the indentation rules, and therefore are very hard to understand. Rewrite it by following the indentation rules (using 3 spaces per indentation level).

```
#include <iostream>
using namespace std;
int main ( )
{
int a;
cout <<”Enter an integer:”; cin >> a;
if (a<0)
{
cout <<”the value needs to be positive\n”;
return 0; }
else { cout <<”the square root is :” << sqrt (a) <<”\n”;
if (a==10)
cout <<” By the way, a’s value is 10\n”; else cout <<”a is not 10\n”;
return 0;
}
}
```