CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 3

Due Sunday, September 23rd, by 6 pm
+5% bonus if submitted by Friday September 21th 11:55 pm,
+2% bonus if submitted by Saturday September 22nd, 11:55 pm

This assignment is due **September 23rd, by 6 pm**

- ***All 4 components*** **(Cloud9 workspace, Moodle CodeRunner quiz attempts, Moodle HW3 MCQs, and zip file) must be completed and submitted by Sunday, September 23rd, 6:00 pm for your homework to receive points.**
- Complete submissions (Cloud9 workspace, Moodle CodeRunner quiz attempts, Moodle HW3 MCQs, and zip file) before **Friday September 21th 11:55 pm** will receive a 5% bonus, and complete submissions before **Saturday September 22th 11:55 pm** will receive a 2% bonus.

Objectives:

- Understand and work with **if-else** conditionals and **switch** statements.
- Writing and testing C++ functions
  - Understand problem description
  - Design your function:
    - come up with a step by step algorithm,
    - convert the algorithm to pseudocode
    - imagine many possible scenarios and corresponding sample runs or outputs
  - Convert the pseudocode into a program written in the C++ programming language
  - Test it in the Cloud9 IDE and submit it for grading on Moodle

---

Develop in Cloud9: For this assignment, write and test your solution using Cloud9.

Submission: All three steps must be fully completed by the submission deadline for your homework to receive points. Partial submissions will not be graded.

1. ***Share your Cloud 9 workspace with your TA:*** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline*.
   - Create a directory called **Hmwk3** and place all your file(s) for this assignment in this directory.
   - Share your workspace by clicking Share in the upper right hand corner and inviting your TA using their Cloud9 username.

TA Name	Cloud9 Username	TA Name	Cloud9 Username
Gabriel Andrade	gpandrade	Varsha Koushik	varshak
Paramjot Singh	param17	John Klingner	john.klingner@colorado.edu
Tetsumichi(Telly) Umada	tetsumichiumada	Sebastian Laudenschlager	slaudens
Richard Tillquist	Carter	Dixit Patel	dixitpatel
Ashwin Sankaralingam	ashwinsankaralingam	Harshini Muthukrishnan	harshini95
Shudong Hao	shudong	Lucas Hayne	luchayne
Andrew Altomare	andrewaltomare	Chu Sheng Ku	chusheng
Isabella Huang	isabellahuang4	Supriya M. Naidu	supriyanaidu

- Make sure to *save* the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
- The file(s) should have all of your functions, test cases for the functions in main function(s), and adhere to the style guide. Please read the **Test Cases** and **Style and Comments** sections for more details.

2. ***Submit to the Moodle Autograder:*** Head over to Moodle to the link **Hmwk 3 CodeRunner**. You will find one programming quiz question for each problem in the assignment. Submit your solution for the first problem and press the Check button. You will see a report on how your solution passed the tests, and the resulting score for the first problem. You can modify your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date. Continue with the rest of the problems.

3. ***Complete the quiz*** **Hmwk 3 MCQ** on Moodle.

4. ***Submit a zip file to Moodle:*** After you have completed all the questions and checked them on Moodle, ***you must submit a zip file with the .cpp file(s) you wrote in Cloud9***. Submit this file going to **Hmwk 3 (File Submission)** on moodle.

Style and Comments (10 points)

Comments (5 points):

- Your code should be well commented. Use comments to explain what you are doing, especially if you have a complex section of code. These comments are intended to help other developers understand how your code works. These comments should begin with two backslashes (//).

- Please also include a comment at the top of your solution with the following format:

```
// CS1300 Fall 2018
// Author:
// Recitation: 123 – Favorite TA
// Cloud9 Workspace Editor Link: https://ide.c9.io/…
// Homework 3 - Problem # ...
```

Algorithm (5 points):

- Remember to include in comments, before the function definition, a description of the algorithm you used for that function.
- This is an example C++ solution. Look at the code and the algorithm description for an example of what is expected.

```
/**
 * Algorithm: that checks what range a given MPG falls into.
 *     1. Take the mpg value passed to the function.
       2. Check if it is greater than 50.
            If yes, then print "Nice job"
       3. If not, then check if it is greater than 25.
            If yes, then print "Not great, but okay."
       4. If not, then print "So bad, so very, very bad"
 * Parameters: miles per gallon (float type)
 * Output: different string based on three categories of
 *          MPG: 50+, 25-49, and less than 25.
 * Returns: nothing
 */

void checkMPG(float mpg)
{
      if(mpg > 50) // check if the input value is greater than 50
      {
            cout << "Nice job" << endl; // output message
      }
      else if(mpg > 25) //if not, check if is greater than 25
      {
            cout << "Not great, but okay." << endl; // output message
      }
      else // for all other values
      {
            cout << "So bad, so very, very bad" << endl; // output message
      }
}
```

The algorithm description below does not mention in detail what the algorithm does and does not mention what value the function returns. Also, the solution is not commented. This would not receive full credit.

```
/**
 * Checks mpg
 */
void checkMPG(float mpg) {
      if(mpg > 50) {
            cout << "Nice job" << endl;
      }
      else if(mpg > 25) {
                  cout << "Not great, but okay." << endl;
      }
      else {
                  cout << "So bad, so very, very bad" << endl;
      }
}
```

Test Cases (10 points)

Code compiles and runs (4 points):

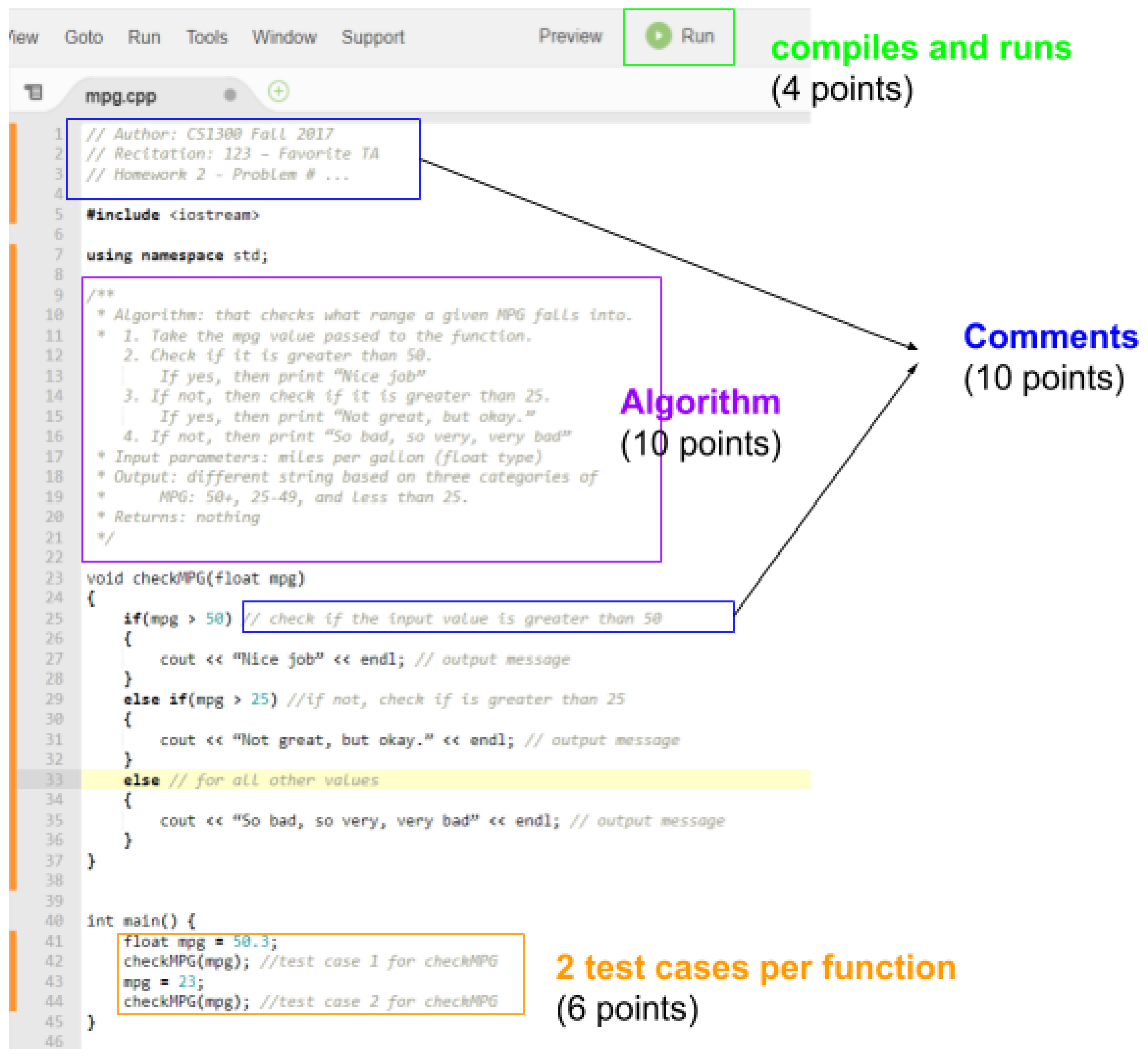
- The zip file you submit to Moodle under **Hmwk3 (File Submission)** should contain one .cpp file for each of the problems in Homework 3. Each .cpp file(s) should be fully functional so that it can be compiled and run on Cloud9 with no errors. Each .cpp file should contain the function (the function code should match the code submitted to the Moodle CodeRunner autograder) and a *main()* for testing (see below).

Test cases (6 points):

- For each function you develop as a solution to one of the problems, you should have at least 2 test cases (function calls) present in the *main()* function. Write as many function calls as needed to ensure your solution works correctly **for all possible values of the input argument(s)** (given the assumptions in the problem description).

The .cpp file(s) you submit should look something like this. Notice that this file has the required comments at the top of the file, an algorithm description, comments within the function, and two tests in main for the function *checkMPG*.

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 3



```cpp
// Author: CS1300 Fall 2017
// Recitation: 123 - Favorite TA
// Homework 2 - Problem # ...

#include <iostream>

using namespace std;

/**
 * Algorithm: that checks what range a given MPG falls into.
 *  1. Take the mpg value passed to the function.
    2. Check if it is greater than 50.
        If yes, then print "Nice job"
    3. If not, then check if it is greater than 25.
        If yes, then print "Not great, but okay."
    4. If not, then print "So bad, so very, very bad"
 * Input parameters: miles per gallon (float type)
 * Output: different string based on three categories of
 *      MPG: 50+, 25-49, and less than 25.
 * Returns: nothing
 */

void checkMPG(float mpg)
{
    if(mpg > 50) // check if the input value is greater than 50
    {
        cout << "Nice job" << endl; // output message
    }
    else if(mpg > 25) //if not, check if is greater than 25
    {
        cout << "Not great, but okay." << endl; // output message
    }
    else // for all other values
    {
        cout << "So bad, so very, very bad" << endl; // output message
    }
}


int main() {
    float mpg = 50.3;
    checkMPG(mpg); //test case 1 for checkMPG
    mpg = 23;
    checkMPG(mpg); //test case 2 for checkMPG
}
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 3

Conditional Statements :

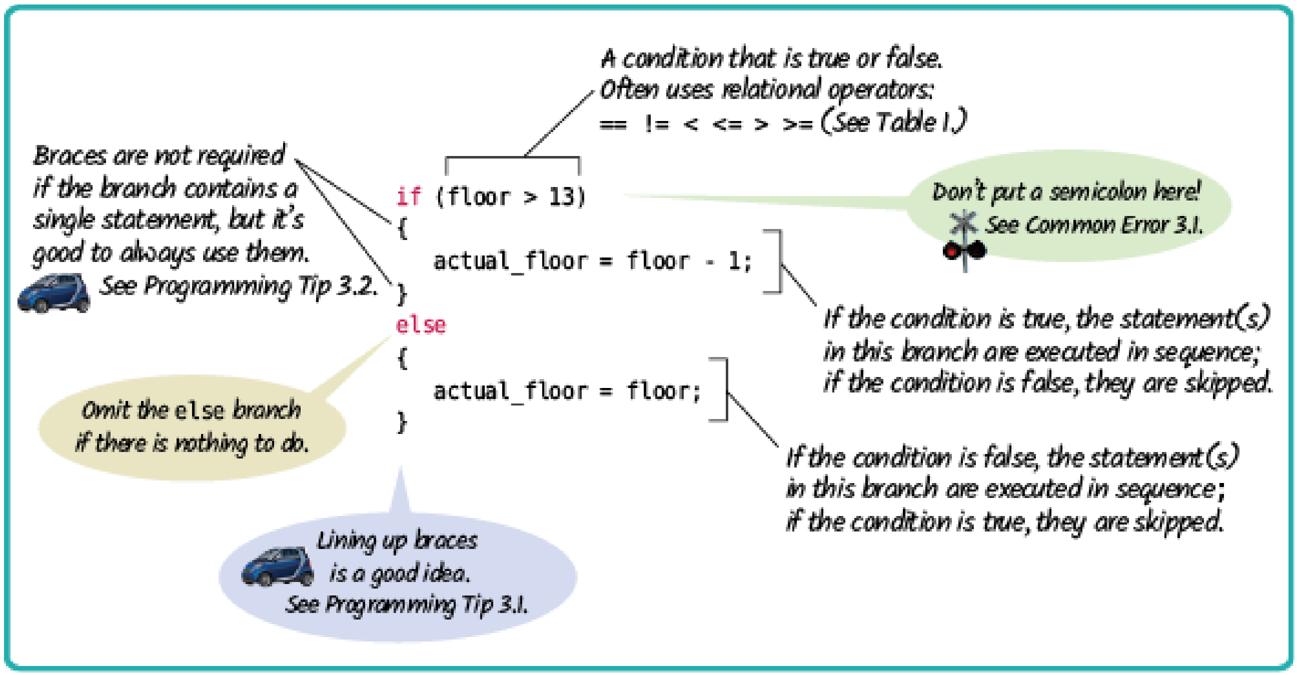
Conditional Statements also known as decision or branching statements, are used to make a decision using a condition. Examples of conditional statements are :

1. If-else statement - used for range as well as exact value branching
2. Switch statement - mostly used for exact value matching

IF-ELSE statements:

The if statement selects and executes the statement(s) based on a given condition. If the condition evaluates to True then a given set of statement(s) is executed. However, if the condition evaluates to False, then the given set of statements is skipped and the program control passes to the statement following the if statement. If we have another else statement next to this, it is handled by the else block.

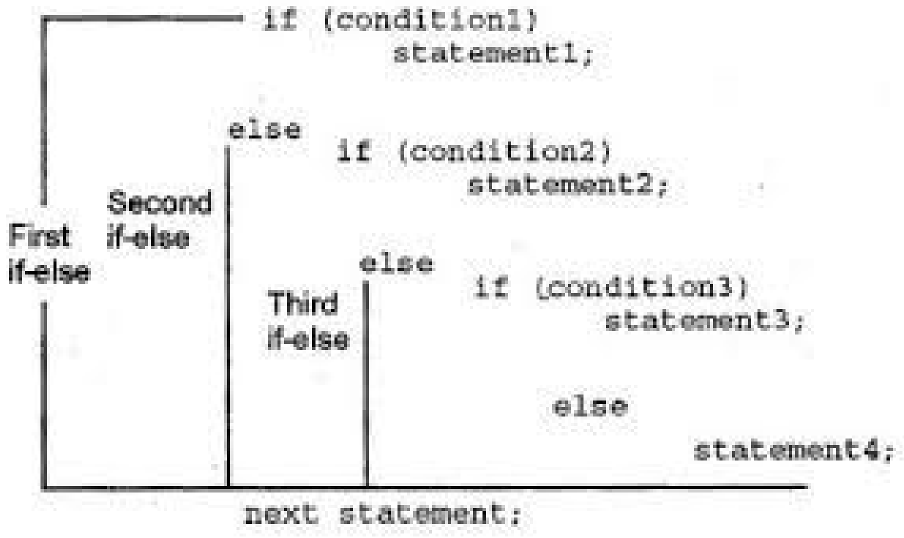
The syntax of the if statement is



```
if (floor > 13)
{
   actual_floor = floor - 1;
}
else
{
   actual_floor = floor;
}
```

A condition that is true or false. Often uses relational operators: == != < <= > >= (See Table 1.)
Braces are not required if the branch contains a single statement, but it's good to always use them. See Programming Tip 3.2.
Don't put a semicolon here! See Common Error 3.1.
If the condition is true, the statement(s) in this branch are executed in sequence; if the condition is false, they are skipped.
Omit the else branch if there is nothing to do.
If the condition is false, the statement(s) in this branch are executed in sequence; if the condition is true, they are skipped.
Lining up braces is a good idea. See Programming Tip 3.1.

Nested if else: When there are more than 2 branches we can use a nested if else conditions.

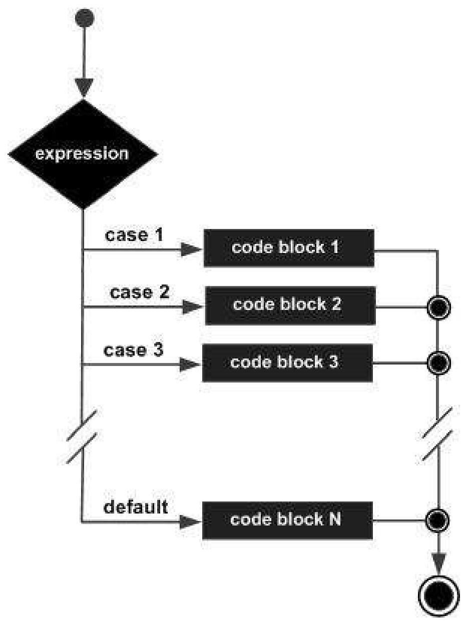


```
if (condition1)
        statement1;
else
    if (condition2)
            statement2;
    else
        if (condition3)
                statement3;
        else
                statement4;
next statement;
```

SWITCH statements:

Switch case statements are a substitute for long if statements that compare a variable to several integral values.

- The switch statement is a multiway branch statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression
- Switch is a control statement that allows a value to change control of execution.



Syntax:

```
switch (n){
      case 1:
            // code to be executed if n = 1;
            break;
      case 2:
            // code to be executed if n = 2
            break;
      default:
            // code to be executed if n   doesn't match any cases
}
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 3

Important notes to keep in mind while using switch statements :

1. The expression provided in the switch should result in a **constant value** otherwise it would not be valid.
    a. switch(23*45) //allowed
    b. switch('a') //allowed (takes the ASCII Value)
    c. switch(a+b) //allowed,where a and b are int variable, which are defined earlier
2. Duplicate case values are not allowed.
3. The **break** statement is used inside the switch to terminate a statement sequence. When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
4. The **default** statement is optional.Even if the switch case statement do not have a default statement,it would run without any problem.
5. Nesting of switch statements are allowed, which means you can have switch statements inside another switch. However nested switch statements should be avoided as it makes program more complex and less readable.
6. The break statement is optional. If omitted, execution will continue on into the next case. The flow of control will fall through to subsequent cases until a break is reached.

Problems Set:
Write a **program** for each of the following problems. You are encouraged to write your solution first in Cloud9. For each of problem, you are required to submit a .cpp file with your function and a main() function with 2 tests cases.

Problem 1 : (15 points)

Write a function **collatzStep** which takes a single integer argument and returns an integer. The return value should be the next value in the Collatz sequence based on the value of the input parameter. If the given value `n` is even, the next value should be `n/2` . If `n` is odd, the next value should be `3n+1` . If the given value is not positive the function should return 0.

- Your function should be named **collatzStep**
- Your function takes one input argument: an **integer** number
- Your function must return an **integer**, as specified above

Examples:
1. If n is 4, the next number in the Collatz sequence is 2
2. If n is 7, the next number in the Collatz sequence is 22
3. If n is -5, the function should return 0

Problem 2:(20 points)

Write a function **countDigits** that takes integer as input and returns how many digits the number has. Assume that all integers are less than 1000 and greater than -1000. Suggestion: if the number is negative, you could first multiply it with –1.

- Your function should be named **countDigits**
- Your function takes one input argument: an **integer** number
- Your function must return the number of digits as an **integer**

Examples::

123 has 3 digits
-75 has 2 digits

Note: There are 4 Multiple Choice Questions (MCQs) which refer to this problem.

Problem 3:(25 points)

Write a function **daysOfMonth** which takes a single input argument, the month, as an integer argument . Using the input value, the function will calculate the number of days in a particular month, and print out the number of days.

- Your function should be named **daysOfMonth**
- Your function takes one input argument: the month, as an **integer** *(assume always positive)*
- Your function does not have a return value
- Your function prints/displays/outputs the number of days in the month, in the format specified below

Note: You must use a switch case for this problem. *if/else* statements are not allowed.

The format of the output should be "Month " <month> " has x (or y) days" . If an invalid month number is passed to the function, it should print "Invalid month number"

Example output:
If the function is called with 4 as the input argument:

```
Month 4 has 30 days
```

If the function is called with 3 as the input argument:

```
Month 3 has 31 days
```

If the function is called with 2 as the input argument:

```
Month 2 has 28 or 29 days
```

If the function is called with 14 as the input argument:

```
Invalid month number
```

CSCI 1300 - Starting Computing
Instructor: Fleming
Homework 3

(Extra Credit : 20 point)
Problem 4 : Minivan Doors

A minivan has two sliding doors. Each door can be opened by either a dashboard switch, its inside handle, or its outside handle. Whether a door can be opened with one of these is predicated on whether the doors are locked, whether the child lock is on, and whether the gear shift is in P (the doors cannot be opened if the car is in any other gear). Keep in mind the child lock can only prevent inside handles from opening the doors, whereas the master lock will inhibit all door handles and dashboard switches.

- For example, if the *inside* right handle is pulled, the *outside* left handle is pulled, the child lock is on, the master lock is off, and the gear shift is in P, then *only the left door opens.*

Your task is to simulate a portion of the control software for the vehicle. The input is a sequence of boolean values for the switches and a character value for the gear shift:

- Dashboard switches for left and right sliding door, child lock, and master lock (`false` for off/unlocked or `true` for activated/locked)
- Inside and outside handles on the left and right sliding doors (`false` for closed/not pulled or `true` for pulled)
- The gear shift setting (one of P N D 1 2 3 R).

Print “Left door opens” , “Right door opens” or "Both doors open" as appropriate. If neither door opens, print “Both doors stay closed”.

- Your function must be named **vehicle**
- Your function will receive 9 input arguments in the following order (See prototype on coderunner quiz):
  - Dashboard switch for left door (`bool`)
  - Dashboard switch for right door (`bool`)
  - Child lock (`bool`)
  - Master lock (`bool`)
  - Inside left latch (`bool`)
  - Outside left latch (`bool`)
  - Inside right latch (`bool`)
  - Outside right latch (`bool`)
  - Gearshift (`char`)
- Your function does not return any value