**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming/Gupta**
**Homework 3**

**Due Sunday, February 11th, by 6 pm**
**+5% bonus if submitted by Friday February 9th 11:55 pm,**
**+2% bonus if submitted by Saturday February 10th 11:55 pm**

This assignment is due **Sunday, February 11th 6pm**.

- ***All components*** **(Cloud9 workspace, moodle quiz attempts, and zip file) must be completed and submitted by Sunday, February 11th 6:00 pm for your homework to receive points.**
- Complete submissions (Cloud9 workspace, moodle quiz attempts, and zip file) before **Friday February 9th 11:55 pm** will receive a 5% bonus, and complete submissions before **Saturday February 10th 11:55 pm** will receive a 2% bonus.

---

**Objectives:**

- Understand and work with while loops.
- Understand and work with if-else conditionals.
- Understand and practice using `cout` to display text and variables (their values) and `cin` to take inputs from the user at run time.
- 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 |
|---|---|---|---|
| Akriti Kapur | akritikapur | Monika Tak | mtak |
| Arcadia(Xiaozhe) Zhang | arcadiaz | Paramjot Singh | param17 |
| Ashwin Sankaralingam | ashwinsankaralingam | Richard Tillquist | riti4538 |
| Bu Sun Kim | busun | Sayali Sonawane | sayalisonawane |
| Chelsea Chandler | chelseachandler | Yichen Wang | wangyichen5151 |
| Harshini Muthukrishnan | harshini95 | | |

- 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**. 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. ***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 (20 points)

*Comments* (10 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 Spring 2018
// Author:
// Recitation: 123 – Favorite TA
// Cloud9 Workspace Editor Link: https://ide.c9.io/…
// Homework 3 - Problem # ...
```

**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming/Gupta**
**Homework 3**

*Algorithm* (10 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 to problem 5 of Homework 1. 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 following algorithm description 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):

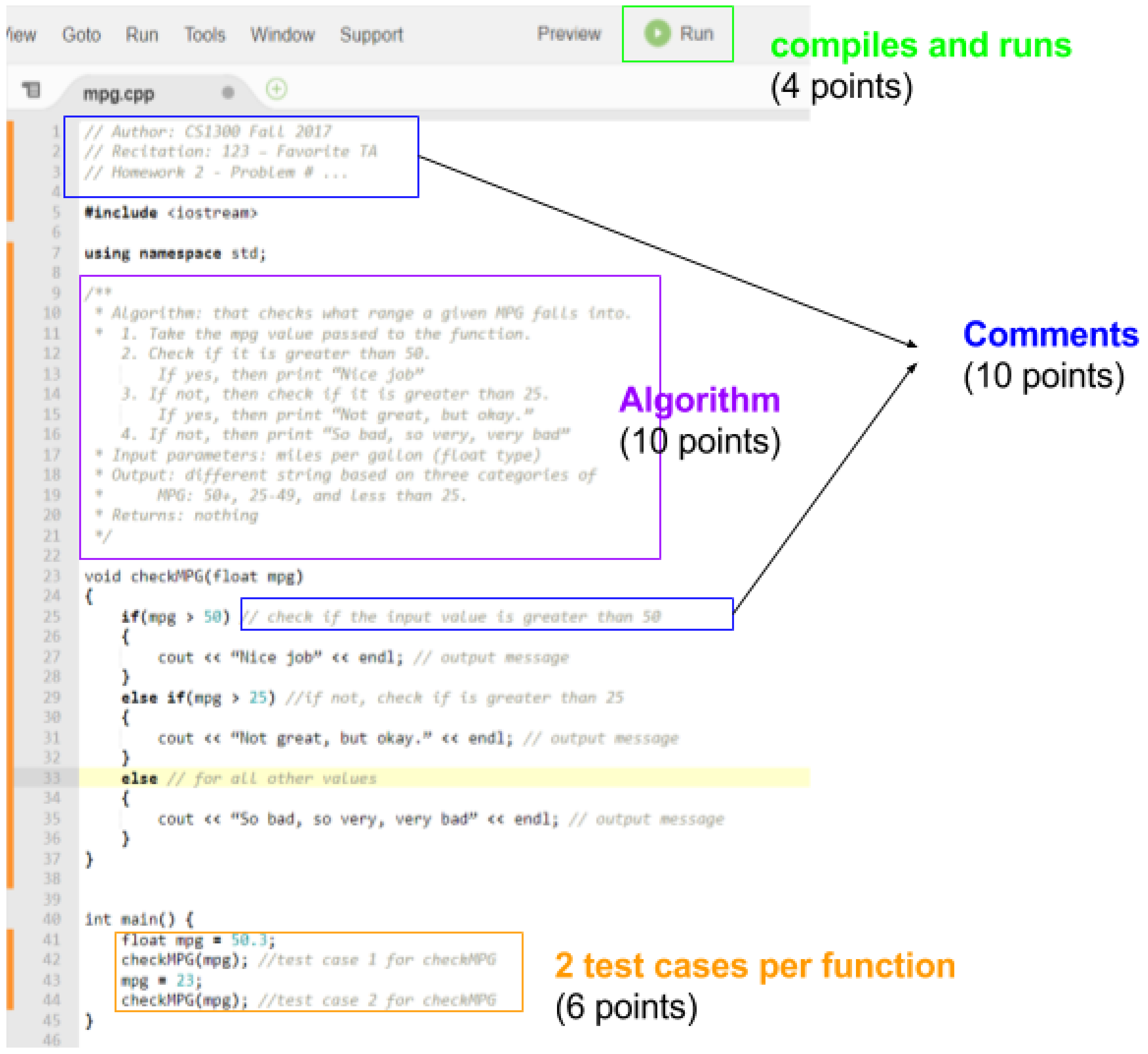
- The zip file you submit to moodle should contain .cpp file(s) that can be compiled and run on Cloud9 with no errors. The functions should match those submitted to the autograder.
- **For this assignment you will be creating two separate .cpp files for each problem. Name the files as problem1.cpp and problem2.cpp.**

*Test cases* (6 points):

- Each function should have 2 test cases present in your main function(s). You may have a single function call to each of the story functions and the menu function in your main as these function do not have any parameters or return values.

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 - Intro to Computer Programming**
**Instructor: Fleming/Gupta**
**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 - Intro to Computer Programming**
**Instructor: Fleming/Gupta**
**Homework 3**

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

## <u>Problem 1: The Story generator</u>

In the game Mad-libs, players are asked for parts of speech, such as noun, adjective, or adverb, and those words are plugged into a template sentence to generate a sometimes-funny story.

### Part A: Story Functions

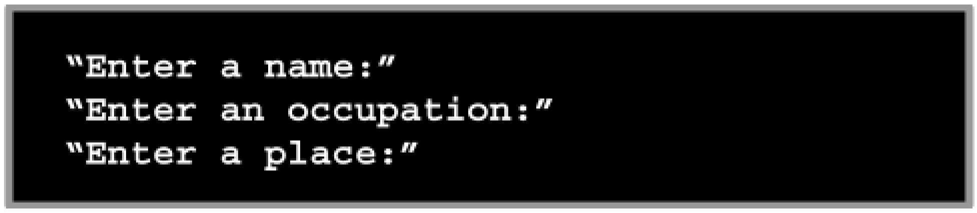
Write 3 functions named **story1**, **story2**, and **story3** that each allow the user to play a simple game of Mad-libs. Each function should store the corresponding story template and ask the user for the missing word types to fill in the template.

- Your functions must be named **story1, story2, and story3.**
- All the three functions **do not return** any value.
- All the three functions **do not take any parameters.**

***Note : In all the following images/examples, <> represents a placeholder.***

For Example, for a story template that looks like this:
<NAME> is a <OCCUPATION> who lives in <PLACE>.

The story function should ask for inputs in the following format:

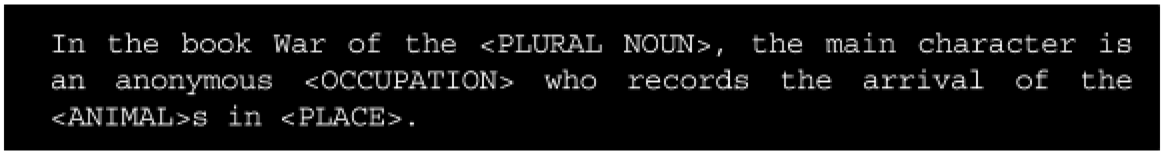


```
"Enter a name:"
"Enter an occupation:"
"Enter a place:"
```

Then the story function should print the new sentence with the user entered inputs in place of the word-type placeholders.

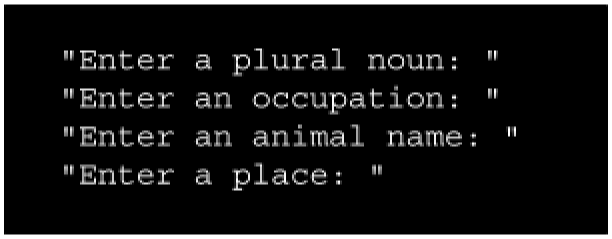
```
Pat is a student who lives in Boulder.
```

**Use the following template for *story1*:**



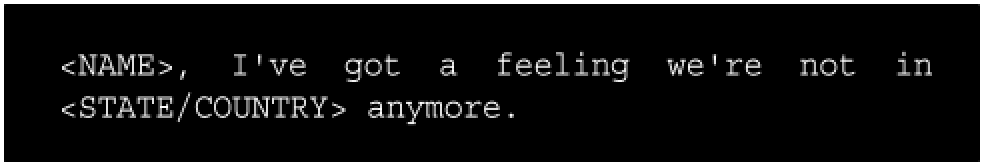
```
In the book War of the <PLURAL NOUN>, the main character is
an anonymous <OCCUPATION> who records the arrival of the
<ANIMAL>s in <PLACE>.
```

**Ask for the following inputs for story1:**



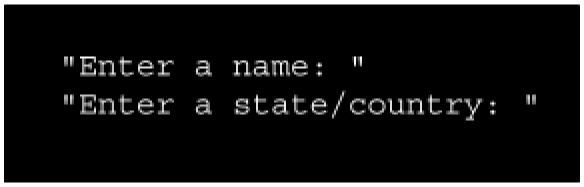
```
"Enter a plural noun: "
"Enter an occupation: "
"Enter an animal name: "
"Enter a place: "
```

**Use the following template for story2:**



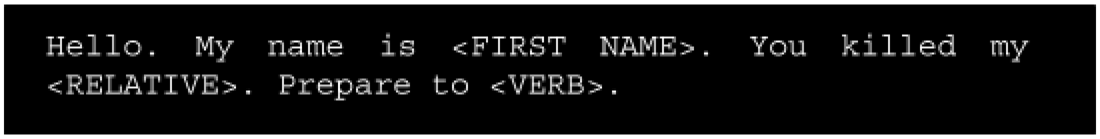
```
<NAME>, I've got a feeling we're not in
<STATE/COUNTRY> anymore.
```

**Ask for the following inputs for story2:**



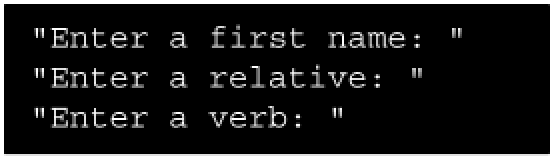
```
"Enter a name: "
"Enter a state/country: "
```

**Use the following template for story3:**



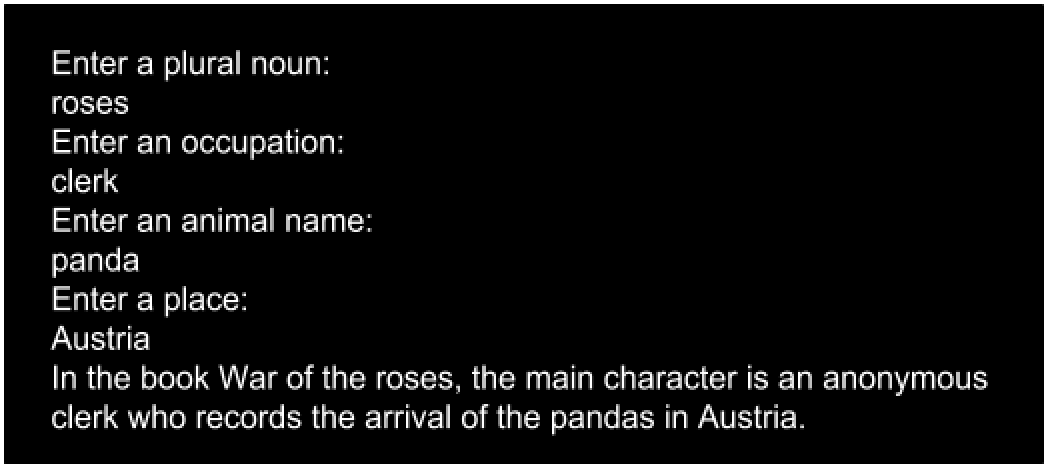
```
Hello. My name is <FIRST NAME>. You killed my
<RELATIVE>. Prepare to <VERB>.
```

**Ask for the following inputs for story3:**



```
"Enter a first name: "
"Enter a relative: "
"Enter a verb: "
```

*For instance, a function call to story1() function would look like the following.*



```
Enter a plural noun:
roses
Enter an occupation:
clerk
Enter an animal name:
panda
Enter a place:
Austria
In the book War of the roses, the main character is an anonymous
clerk who records the arrival of the pandas in Austria.
```

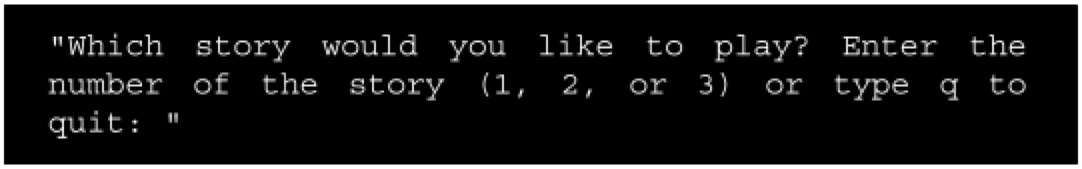
*Note: Test each of the story functions by calling them from your main function.*

## Part B : Menu Function

Write a function called **menu** that is called by main when your program starts.

- Your function must be named **menu**
- Your function **does not return** any value.
- Your function **does not** take any **parameters.**
- Your function must call **story1, story2 and story3** functions.

Inside your **menu** function, you should first ask the user which story they want to play. Your question should look like:



```
"Which story would you like to play? Enter the
number of the story (1, 2, or 3) or type q to
quit: "
```

- If the user types q your program should print "`good bye`" and exit the function.
- If the user types in an invalid input, for example "17" or "slkerjqoe", you should print "`Valid choice not selected.`"
- If the user types in a valid story number, the **menu** function should call the corresponding story function (**story1**, **story2**, or **story3**).

The user should be allowed to play the game as many times as they like. After printing the new story, your program should ask again which story they would like to play.

```
"Which story would you like to play? Enter the
number of the story (1, 2, or 3) or type q to
quit: "
```

*Note: User-input to play further must be obtained inside the* ***menu*** *function, not within your* ***story1****,* ***story2****, and* ***story3*** *functions. That is, the loop that enables the game to be played indefinitely must not be within your story functions*

## <u>Problem 2</u> : <u>Wind Chill</u>

The wind chill is the still-air temperature that would have the same cooling effect on exposed human skin as a given combination of temperature and wind speed.
"Windchill." *Merriam-Webster.com*. Merriam-Webster, n.d. Web. 5 Sept. 2017.

For this problem, we will be recreating part of this wind chill calculator provided by the National Weather Service (NWS).

$$\text{Wind Chill } (^\circ\text{F}) = 35.74 + 0.6215T - 35.75(V^{0.16}) + 0.4275T(V^{0.16})$$

Where, T= Air Temperature (°F) V= Wind Speed (mph) *Effective 11/01/01*

Use this formula for wind chill for your calculations.

### Part A

Write a function **windChillCalculator** to determine the wind chill.

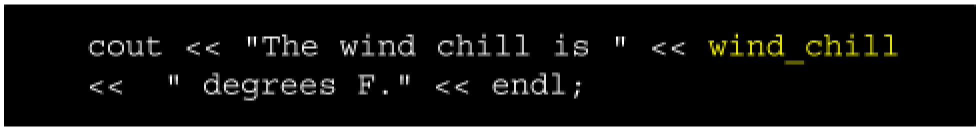
- Your function must be named **windChillCalculator**.
- Your function should take in **T (air temperature)** and **V (wind speed)** as parameters. **Fractional values are allowed.**
- Your function must **return** the wind chill calculated. **Wind chill calculated can be fractional.**

*To test your code, for T = 30.0 and V = 5.0, you should get a Wind Chill of 24.7268.*

Also handle the cases of wrong inputs using if.. else.. statements.
For example, wrong input might be negative value of speed.
In case of wrong input, print the following statement in your function. And return 0 from the function.

```
cout << "Not applicable" << endl;
```

Within your main function, print the following cout statement:



```
cout << "The wind chill is " << wind_chill
<<  " degrees F." << endl;
```

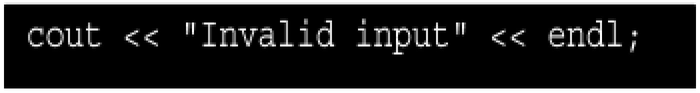
*HINT: Look at the pow() function in the C++ math.h library*

**Part B**

Once you have the wind chill calculation working, create another function **printWindChill**. This function should call **windChillCalculator** and print all the wind chill values for a fixed temperature and variable wind speeds. You will need to write a loop in this function to iterate over the different wind speeds.

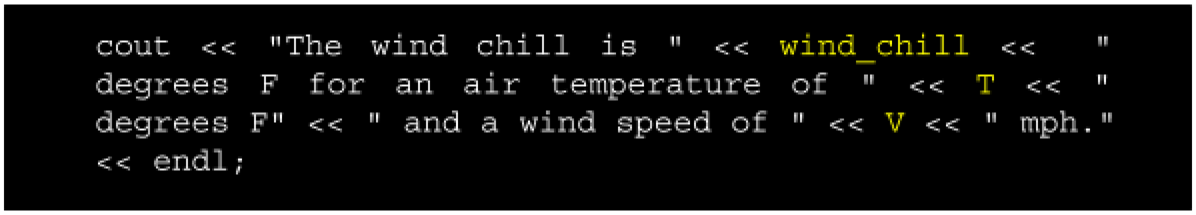
- Your function must be named **printWindChill**.
- Your function should receive **T (air temperature), low_wind_speed, high_wind_speed, and wind_speed_step** as parameters in this order. **All the parameters can be fractional.**
- Your function **does not return** any value.
- This function should call **windChillCalculator** function that you wrote in your previous part to calculate wind chill.

**In case the low wind speed is more than the high wind speed OR the wind_speed_step is a negative value, your function should print the following:**



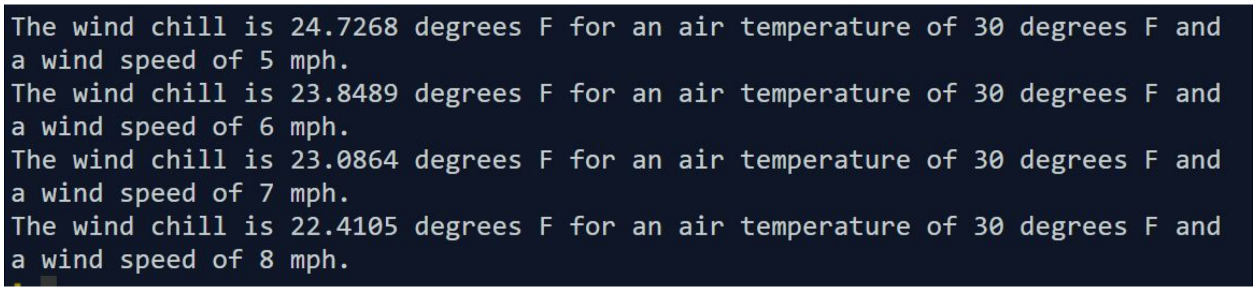
```
cout << "Invalid input" << endl;
```

Use the following cout statement to output the values from within the function:



```
cout << "The wind chill is " << wind_chill << "
degrees F for an air temperature of " << T << "
degrees F" << " and a wind speed of " << V << " mph."
<< endl;
```

For Example: Let the parameters be T = 30.0, low_wind_speed = 5.0, high_wind_speed = 8.0, and wind_speed_step = 1.0, your function should print out the wind chill for a temperature of 30 °F and wind speeds starting from 5 mph and ending at 8 mph, incrementing by 1 mph. (5 mph, 6 mph, 7 mph, and 8 mph):



```
The wind chill is 24.7268 degrees F for an air temperature of 30 degrees F and
a wind speed of 5 mph.
The wind chill is 23.8489 degrees F for an air temperature of 30 degrees F and
a wind speed of 6 mph.
The wind chill is 23.0864 degrees F for an air temperature of 30 degrees F and
a wind speed of 7 mph.
The wind chill is 22.4105 degrees F for an air temperature of 30 degrees F and
a wind speed of 8 mph.
```