

Lab 13: Counter-controlled, Flag-controlled, and Sentinel-controlled loops**Due:** 10/31/24

In this lab you will practice using while loops to perform repetitive tasks. There are many different types of while loops. In this lab you will review and write three functions that each demonstrate a different type of while loop.

Function main():

Just calls the three functions described below. For each of them, it:

- 1) Calls the function.
- 2) Pauses the program to show the results.
- 3) Clears the screen (except after the last call).

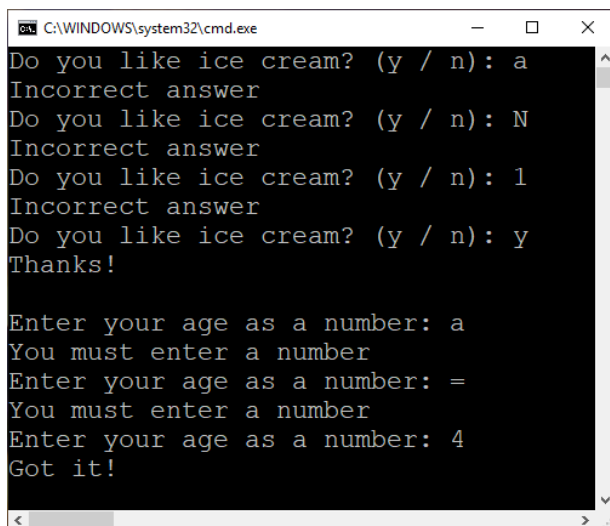
Function flag_controlled_loop():

This function uses a couple of flag-controlled loop to ensure the user enters correct information in two separate situations. In a flag-controlled loop a Boolean variable is used as the loop condition.

For your function you will ask for user input and loop until the user enters valid input.

First ask the user the yes/no question “Do you like ice cream?” and ask them to input y or n as their answer. Check to make sure the input is valid. Then ask the user to enter their age. Remember that, if the user does not enter a number, the input stream goes into the fail state. You need to reset the input stream, clear whatever is on the input stream, and then ask the user to try again. To get the input stream back into working order, use:

```
if ( !cin ){
    cin.clear();
    cin.ignore(2000, '\n');
}
```

Sample run of this function

```
C:\WINDOWS\system32\cmd.exe
Do you like ice cream? (y / n): a
Incorrect answer
Do you like ice cream? (y / n): N
Incorrect answer
Do you like ice cream? (y / n): 1
Incorrect answer
Do you like ice cream? (y / n): y
Thanks!

Enter your age as a number: a
You must enter a number
Enter your age as a number: =
You must enter a number
Enter your age as a number: 4
Got it!
```

Example Program

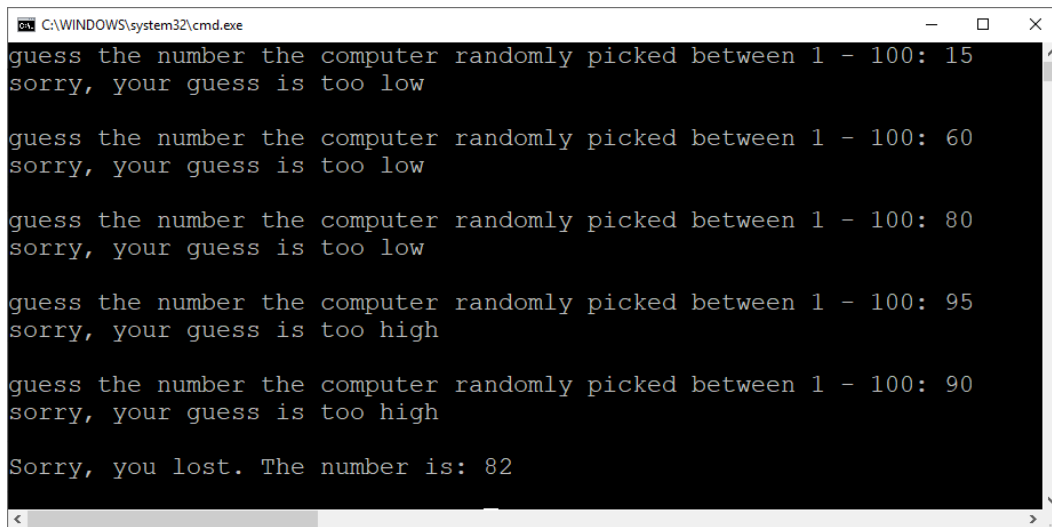
The **flagloop.cpp** program provided illustrates the skills you are learning in this lab. Open it in your IDE, read and understand the code and finally run the program with different inputs to see how it works. Next, implement your function.

Function `counterNflag_controlled_loop()`

This function uses a mix of counter-controlled and flag-controlled loops to implement a guessing game. It generates a random number (1..100) and gives the user five chances to guess it. The user receives feedback from the program regarding whether the guessed number is too high or too low, if it is incorrect. They get a message indicating they win, if the guess is correct.

You will need to use a complex condition for the while loop. One part of the complex condition is whether the user has guessed the number, and the other part is whether the user has used all the attempts to guess the number. Complex conditions in while loops are similar to those in if statements. You connect the conditions with a Boolean operator.

Sample runs of this function



```
C:\WINDOWS\system32\cmd.exe
guess the number the computer randomly picked between 1 - 100: 15
sorry, your guess is too low

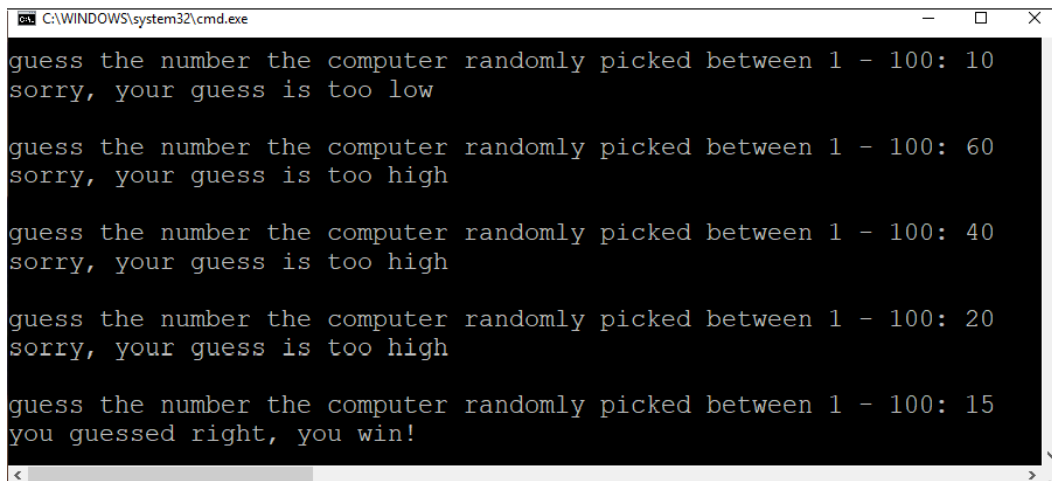
guess the number the computer randomly picked between 1 - 100: 60
sorry, your guess is too low

guess the number the computer randomly picked between 1 - 100: 80
sorry, your guess is too low

guess the number the computer randomly picked between 1 - 100: 95
sorry, your guess is too high

guess the number the computer randomly picked between 1 - 100: 90
sorry, your guess is too high

Sorry, you lost. The number is: 82
```



```
C:\WINDOWS\system32\cmd.exe
guess the number the computer randomly picked between 1 - 100: 10
sorry, your guess is too low

guess the number the computer randomly picked between 1 - 100: 60
sorry, your guess is too high

guess the number the computer randomly picked between 1 - 100: 40
sorry, your guess is too high

guess the number the computer randomly picked between 1 - 100: 20
sorry, your guess is too high

guess the number the computer randomly picked between 1 - 100: 15
you guessed right, you win!
```

Example Program

The example program calculates x to the power of y using a counter-controlled loop. It works similar to the `pow(x, y)` function.

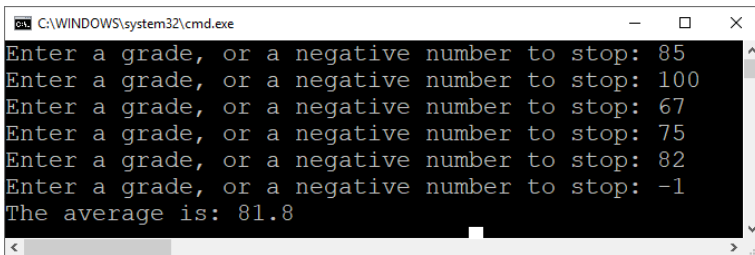
The **xpowy.cpp** program provided illustrates the skills you are learning in this lab. Open it in your IDE, read and understand the code and finally run the program with different inputs to see how it works. Next, implement your function.

Function 3: Sentinel-controlled while Loops

This function uses a sentinel-controlled loop to get whole numbers greater than or equal to zero in order to calculate their average. It keeps asking for a grade until the user enters a negative value and then, it calculates and displays the average. A Sentinel controlled while loop keeps looping until the user enters a specific value. We call this value the sentinel value.

Your function will ask the user to input a series of grades, and to input a negative value (the sentinel) when done. You will take the average of the grades, being careful to not include the sentinel value in the average.

Sample run of this function



```

C:\WINDOWS\system32\cmd.exe
Enter a grade, or a negative number to stop: 85
Enter a grade, or a negative number to stop: 100
Enter a grade, or a negative number to stop: 67
Enter a grade, or a negative number to stop: 75
Enter a grade, or a negative number to stop: 82
Enter a grade, or a negative number to stop: -1
The average is: 81.8
  
```

The program must compile without errors or warnings.

Open **lab13.cpp** in your IDE and implement the algorithms provided in the source code as comments.

I am posting my sample runs for your reference. Please run your program as many times as necessary to ensure that you tested it thoroughly. Make sure it behaves like my sample runs.

Don't forget to include at the top of the programs, the comments shown below with your information (name, class and section number, etc.)

```

////////////////////////////////////
//
// Name: <Put your name here>
// Date: <Today's date>
// Class: <Your class number and section number, like: CSCI 1470.02>
// Semester: <This semester, like: Fall 2012>
// CSCI 1470 Instructor: <Your lecture instructor's name>
//
// Program Description: Enter here your description of what the program does
//
////////////////////////////////////
  
```

When done, submit your solution through Blackboard using the “Assignments” tool. Do Not email it.

Paste the **link** to your final solution along with your **source code** in the textbox opened when you click on **Create Submission** before you click on **Submit**.

The following is the basic criteria to be used to grade your submission:

You start with 100 points (the first two programs are worth 33 points while the last one is worth 34 points) and then lose points as you don't do something that is required.

Function 1: flag_controlled_loop()

- 30: Incorrect/missing implementation of the flag_controlled_loop() function
- 10: Loop 1: Doesn't loop until user enters either 'y' or 'n'
- 10: Loop 2: Doesn't loop until user enters a number

Function 2: counterNflag_controlled_loop()

- 30: Incorrect/missing implementation of the counterNflag_controlled_loop() function
- 10: Doesn't stop looping if guess is correct, or after 5 guesses
- 10: Doesn't correctly determine win or lose

Function 3: sentinel_controlled_loop()

- 30: Incorrect/missing implementation of the sentinel_controlled_loop() function
- 10: Doesn't stop looping when user enters a negative value
- 5: Doesn't correctly take sum of user input (must exclude the negative value)
- 5: Doesn't correctly calculate average of user input (integer division or other problems)

-30: Program doesn't compile

-5: Missing comments at the top of the program

-20: Program does not implement the provided algorithm (each function)

-10: Incorrect/missing function call (each function)

-20: Incorrect/missing source code

-20: or incorrect/missing link to your solution

-100: The code submitted is not your creation (you got it from a web site or another person)

-10: Late

Important: more points may be lost for other reasons not specified here.