Coding Assignment 4

CSE 1310

Fall 2023

1. The format and content of the output is not a suggestion – it is the specification given to you to follow so please follow it exactly. Points will be lost for not following the specification.
2. Please note that part of the rubric is **how** you coded the program in addition to outputting valid responses.
3. Make sure your name and student id are in a comment at the top of your program.
4. Name your program `Code4_studentid.c` for submission to Canvas - only submit the .zip file.
5. Check the rubric BEFORE submitting to ensure that you have fulfilled all requirements.
6. Make sure your code is warning and error free before submitting.

Problem Statement

You will create a program to print the output of a triple nested for loop based on input values.

Required Code Elements

Your program will contain 3 functions – `main()`, `GetValue()` and `CheckValues()`. You are not required to use the exact names of `GetValue()` and `CheckValues()` but this document will refer to them by those names.

The following pseudocode should be used to create your program.

Function `main()`

Prompt for which character to use for the output – this character should be stored in a single `char` variable.

Using a `do-while`

Call function `GetValue()` with parameters “starting” and “outer”. Store the return value as the starting counter value of the outermost for loop.

Call function `GetValue()` with parameters “ending” and “outer”. Store the return value as the ending counter value of the outermost for loop.

The while condition should call function `CheckValues()` with two parameters – the starting counter value returned by the first call to `GetValue()` and the ending counter value returned by the second call to `GetValue()`. If `CheckValues()` returns true, then the do-while should continue and prompt for new values (since the input values were invalid).

Using a do-while

Call function `GetValue()` with parameters “starting” and “middle”. Store the return value as the starting counter value of the middle for loop.

Call function `GetValue()` with parameters “ending” and “middle”. Store the return value as the ending counter value of the middle for loop.

The while condition should call function `CheckValues()` with two parameters – the starting counter value returned by the first call to `GetValue()` and the ending counter value returned by the second call to `GetValue()`. If `CheckValues()` returns true, then the do-while should continue and prompt for new values (since the input values were invalid).

Using a do-while

Call function `GetValue()` with parameters “starting” and “inner”. Store the return value as the starting counter value of the innermost for loop.

Call function `GetValue()` with parameters “ending” and “inner”. Store the return value as the ending counter value of the innermost for loop.

The while condition should call function `CheckValues()` with two parameters – the starting counter value returned by the first call to `GetValue()` and the ending counter value returned by the second call to `GetValue()`. If `CheckValues()` returns true, then the do-while should continue and prompt for new values (since the input values were invalid).

Create a triple nested for loop

for (starting value of outermost loop -> ending value of outermost loop

for (starting value of middle loop -> ending value of middle loop)

for (starting value of innermost loop -> ending value of innermost loop)

print input character

Add newlines as needed to obtain proper output (see output examples)

Function `CheckValues()`

Returns 1 for true or 0 for false and takes two parameters – starting value and ending value

If starting value is greater than ending value

Print message about invalid input

Return 1 for true

Else

Return 0 for false

Function `GetValue`

Returns the prompted for value and takes two parameters – two strings where the first string is “starting” or “ending” and the second string is “outer” or “middle” or “inner”. These strings are used to print the appropriate prompt for getting input. See output examples. Function returns the input value.

Required Program Output

Example of all valid input with * as the character to be printed.

```
Enter the character you want to use *
Enter the starting value for the outer loop 1
Enter the ending value for the outer loop 3
Enter the starting value for the middle loop 2
Enter the ending value for the middle loop 5
Enter the starting value for the inner loop 3
Enter the ending value for the inner loop 7

****
****
****

****
****
****
```

Example of invalid input for the middle loop with & as the character to be printed.

```
Enter the character you want to use &
Enter the starting value for the outer loop 4
Enter the ending value for the outer loop 6
Enter the starting value for the middle loop 2
Enter the ending value for the middle loop 1

Starting value must be less than ending value. Please reenter

Enter the starting value for the middle loop 1
Enter the ending value for the middle loop 7
Enter the starting value for the inner loop -1
Enter the ending value for the inner loop 4

&&&&&
&&&&&
&&&&&
&&&&&
&&&&&
&&&&&

&&&&&
&&&&&
&&&&&
&&&&&
&&&&&
&&&&&
```

Example of invalid input for the outer loop with @ as the character to be printed.

```
Enter the character you want to use @
Enter the starting value for the outer loop 0
Enter the ending value for the outer loop -1

Starting value must be less than ending value. Please reenter

Enter the starting value for the outer loop -8
Enter the ending value for the outer loop -5
Enter the starting value for the middle loop 3
Enter the ending value for the middle loop 7
Enter the starting value for the inner loop 2
Enter the ending value for the inner loop 9

@@@@@@@
@@@@@@@
@@@@@@@
@@@@@@@

@@@@@@@
@@@@@@@
@@@@@@@
@@@@@@@

@@@@@@@
@@@@@@@
@@@@@@@
@@@@@@@
```

```
Enter the character you want to use ^
Enter the starting value for the outer loop 4
Enter the ending value for the outer loop 3

Starting value must be less than ending value. Please reenter

Enter the starting value for the outer loop 2
Enter the ending value for the outer loop 5
Enter the starting value for the middle loop 4
Enter the ending value for the middle loop 1

Starting value must be less than ending value. Please reenter

Enter the starting value for the middle loop 1
Enter the ending value for the middle loop 4
Enter the starting value for the inner loop 6
Enter the ending value for the inner loop 5

Starting value must be less than ending value. Please reenter

Enter the starting value for the inner loop 5
Enter the ending value for the inner loop 7

^^
^^
^^

^^
^^
^^

^^
^^
^^
```

Testing

Run your program multiple times.

Enter invalid values (starting value is less than ending value) for each of layer of the for loops – inner, middle and outer – to ensure that the validity check is working for each one.

Enter both positive and negative values for your starting and ending values. Negative values should work just fine as long as the starting value is less than the ending value.

Note the relationship between the outermost loop range, the middle loop range and the innermost loop range. The outermost loop range determines the number of blocks displayed. The middle loop range determines the number of rows per block. The innermost loop range determines the number of characters displayed per row. Make sure you understand this relationship and how the code forms the output. For an OLQ, you will be given the output of a triple nested for loop and asked to write the for loop that created it.

Hints

The questions/prompts go in the `GetValue()` function - you will use the two passed in strings to form the question. You are passing a quoted string to the function from `main()`.

This allows the function to produce different questions based on what strings are passed. That's why GetValue() is called six times from `main()` - you pass in 6 different combination of strings to get the six different prompts shown in the assignment's output. Each time you call it, it passes back the value it prompted for.

Two calls to `GetValue()` go into each of the three do-while.

If you had this for loop...

```
for (int i = 0; i < 5; i++)
```

then the starting value of the loop is what i is initialized to (which is 0) and the end value of the loop is 5 (loop stops when i reaches 5)

If you had to start this loop at 4 and end at 1234, then it would be

```
for (int i = 4; i < 1234; i++)
```

See what changed? The initial value of i is the starting value and the test of i is the ending value.

In this assignment, you still need loop counters/iterators - you can't do it without an i, j and k (for example). You are just substituting variables for the starting value (0 and 4 in these examples) and the ending value (5 and 1234 in these examples).

Don't use your starting and ending values as counters/iterators.