Programming Exercise 7.2

Counting Backwards

Purpose. Learn how to write a reverse-counting loop. It also get you to use "sleep code" that causes a program to pause for a specified period of time. Also learn that "sleep code" is different between Windows and Mac, and how to write code that works on either system. Finally, learn about `'\r'` -- the "carriage return" symbol -- and what it does in console output.

Requirements. Write **`countdownTimer.cpp`**, to count down from 10 to 0, one number at a time, pausing one second per number. The program should take 10 seconds to run, starting with the 2-digit output `"10"`, overwriting it with `" 9"`, and then `" 8"`, etc, until it reaches `" 0"` and ends. Then print some statement on the next line after " 0" is reached, such as "blast off".

NOTE: "Overwrite" means that each line of output should erase the previous one, just as in the odometer example from chapter 7. The `\r` and the flush statement accomplish this.

NOTE: It should *not* be 09, etc. The 10 should be replaced by a space and a 9. Not a 0 and a 9, and not a 9 and a space. Don't overthink this -- there are several ways to do it, including formatting, or if-else structures, etc.

Do *not* include extra text or spaces in the output. The first "10" should be fully left-justified with no spaces of text before it or after it. The "9" through "0" should be printed with a single space in front, only -- nothing else before or after it.

Your program should compile and run in both g++ and Visual Studio. You do not have to actually compile in both, but you do have to use the `#ifdef` code blocks from chapter 7's `digitalClock.cpp` to accomplish this.

Program I/O. Input: none. Output: A series of numbers overwriting one another on the same line of the console screen, and an ending "blast off".

Examples. Here's what the output should look like:

When it starts…
```
10
```

When it ends…
```
 0 blast off!
```

Programming Exercise 7.2

Counting Backwards

Purpose. Learn how to write a reverse-counting loop. It also get you to use "sleep code" that causes a program to pause for a specified period of time. Also learn that "sleep code" is different between Windows and Mac, and how to write code that works on either system. Finally, learn about `'\r'` -- the "carriage return" symbol -- and what it does in console output.

Requirements. Write **`CountdownTimer.java`**, to count down from 10 to 0, one number at a time, pausing one second per number. The program should take 10 seconds to run, starting with the 2-digit output `"10"`, overwriting it with `" 9"`, and then `" 8"`, etc, until it reaches `" 0"` and ends. Then print some statement on the next line after " 0" is reached, such as "blast off".

NOTE: "Overwrite" means that each line of output should erase the previous one, just as in the odometer example from chapter 7. The `\r` and the flush statement accomplish this.

NOTE: It should *not* be 09, etc. The 10 should be replaced by a space and a 9. Not a 0 and a 9, and not a 9 and a space. Don't overthink this -- there are several ways to do it, including formatting, or if-else structures, etc.

Do *not* include extra text or spaces in the output. The first "10" should be fully left-justified with no spaces of text before it or after it. The "9" through "0" should be printed with a single space in front, only -- nothing else before or after it.

Program I/O. Input: none. Output: A series of numbers overwriting one another on the same line of the console screen, and an ending “blast off”.

Examples. Here's what the output should look like:

When it starts…
```
10
```

When it ends…
```
 0 blast off!
```

Programming Exercise 7.2

Counting Backwards

Purpose. Learn how to write a reverse-counting loop. It also get you to use "sleep code" that causes a program to pause for a specified period of time. Also learn that "sleep code" is different between Windows and Mac, and how to write code that works on either system. Finally, learn about `'\r'` -- the "carriage return" symbol -- and what it does in console output.

Requirements. Write **`countdownTimer.py`**, to count down from 10 to 0, one number at a time, pausing one second per number. The program should take 10 seconds to run, starting with the 2-digit output `"10"`, overwriting it with `" 9"`, and then `" 8"`, etc, until it reaches `" 0"` and ends. Then print some statement on the next line after " 0" is reached, such as "blast off".

NOTE: "Overwrite" means that each line of output should erase the previous one, just as in the odometer example from chapter 7. The `\r` and the flush statement accomplish this.

NOTE: It should *not* be 09, etc. The 10 should be replaced by a space and a 9. Not a 0 and a 9, and not a 9 and a space. Don't overthink this -- there are several ways to do it, including formatting, or if-else structures, etc.

Do *not* include extra text or spaces in the output. The first "10" should be fully left-justified with no spaces of text before it or after it. The "9" through "0" should be printed with a single space in front, only -- nothing else before or after it.

Your program should compile and run in both g++ and Visual Studio. You do not have to actually compile in both, but you do have to use the `#ifdef` code blocks from chapter 7's `digitalClock.py` to accomplish this.

Program I/O. Input: none. Output: A series of numbers overwriting one another on the same line of the console screen, and an ending “blast off”.

Examples. Here's what the output should look like:

When it starts…
```
10
```

When it ends…
```
 0 blast off!
```