Due Sunday 10/17/2021 by 9:00pm via Cougar Courses

As you read the following OLI pages and complete the interactive activities, capture the screenshots of the completed activities and replace the respective screenshots in the document.

- Page 29 Counter-based while loops
- Page 30 Loop-based analyses
- Page 31 Non-counter based while loops
- Page 32 The do ... while loops

When you are ready to submit the assignment, download the document in PDF and submit the PDF file on Cougar Course as the proof for your work.

Page 29 Counter-based while loops

LBD counting up for 10 push-ups

Answer the following questions to dissect the process of using a counter to help us complete 10 push-ups.

Before we start exercising, how many push-ups have we done? 0

✔ Correct! There's no push-up before we start exercising.

When should we update the counter?

- ○ Before completing a push-up
- ◉ After completing a push-up

✔ Correct; we earn it.

Our target is 10 push-ups. Should we do another push-up when the counter is 5?

- ◉ Yes
- ○ No

✔ Correct, you're half way there, keep it up.

Mark all of the counter values indicating that we should still do more push-up.

- ☑ 8
- ☑ 9
- ☐ 10

Check My Answer

✔ Correct, keep going, you are almost there.

In summary, you should do another push-up as long as the following comparison evaluates to true.

- ◉ counter < 10
- ○ counter <= 10
- ○ counter > 10

✔ Correct, you may stop as soon as the counter is 10.

How should we update the counter as we make progress?

- ○ counter -= 1;
- ◉ counter += 1;

✔ Correct, this will get us closer to the target.

MR counter update

Why do you think the statement `counter += 1;` must be included inside the loop block?

In order to continue adding the number until it's reached limit

Resubmit

✔ Thanks for sharing.

Hotspot common loop vocabulary

Given the following code segment

```
1 int countdown = 10;
2 while (countdown > 0) {
3   cout << "*";
4   countdown --;
5 }
```

Drag choices to the correct targets in the table

int countdown = 10;	Initialization
countdown > 0	Loop condition
{cout << "*"; countdown --;}	Loop body

✔ Correct. Loop body is what needs to be repeated.

Lines 3 & 4 are executed when `countdown > 0` evaluates to

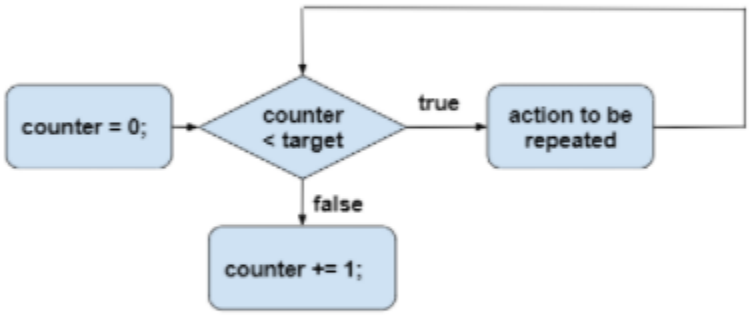
- ○ false
- ◉ true

✔ Correct.

LBD One True Brace Style

```
1  int counter = 0;
2  int target = 10;
3  while (counter < target)
4    cout << "Down ... hold ... UP\n";
5    counter += 1;
```

This is reflected by the following diagram.



A diagram for the above code segment

If we had used the above program as our exercise companion, how many push-ups would we need to do before the program stops telling us to do another push-up? forever

✔ Correct, since counter is never updated inside the loop, the loop condition will never be false.

LBD counting down for 10 push-ups

If our target is to do 10 push-ups, before we start, how many push-ups are there left to do?

10

✔ Correct! We have not done any and therefore need to do 10 more push-ups.

When should we update the countdown?

- ○ Before completing a push-up
- ◉ After completing a push-up

✔ Correct; we earn it.

How should we adjust the countdown as we make progress with our push-ups?

- ◉ countdown -= 1;
- ○ countdown += 1

✔ Correct, this helps reduce the push-ups left in the exercise.

If our target is 10 push-ups, should we do another one when the countdown is 5?

- ○ No
- ◉ Yes

✔ Correct, you're half way there, keep it up.

Mark all of the countdown values indicating that we should still do another one.

- ☑ 2
- ☑ 1
- ☐ 0

Check My Answer

✔ Correct, keep going, you are almost there.

In summary, you should do another push-up as long as the following comparison evaluates to `true`.

- ○ countdown < 10
- ◉ countdown > 0
- ○ countdown <= 10
- ○ countdown >= 0

✔ Correct. you may stop as soon as you have no more left to do.

MR loop ideas

How are you feeling about the counter-based while loops?

Good.

Resubmit

✔ Thanks for your feedback. ×

The simple structure of `while` loops gives developers a lot of flexibility and the freedom to implement creative solutions. What are some problems you think could be solved using while loops?

Running laps.

Resubmit

✔ ×

Hotspot a counter-based loop

Click on the hotspots for tips about how the above loop has been set up to process multiple employees.

Click on this link to access the program in Repl.it.

Run the program by entering 0 as the number of people you would like to process.

How many times did the statements inside the loop body execute? 0

✔ Correct! Since target is 0, counter < target is false, prohibiting the loop body be executed.

Run the program by entering 1 as the number of people you would like to process.

What would be the value of counter after the loop was completed? 1

✔ Correct! Since target is 1, the loop was completed when counter is the same as target.

Run the program by entering 2 as the number of people you would like to process.

What would be the value of counter after the loop was completed? 2

✔ Correct! Since target is 2, the loop was completed when counter is the same as target.

Run the program with your desired number of people for the `target` variable.

Would it be possible for the loop to collect an employee's hourly rate and hours worked without either calculating the wage for the employee or display the wage? Why or why not?

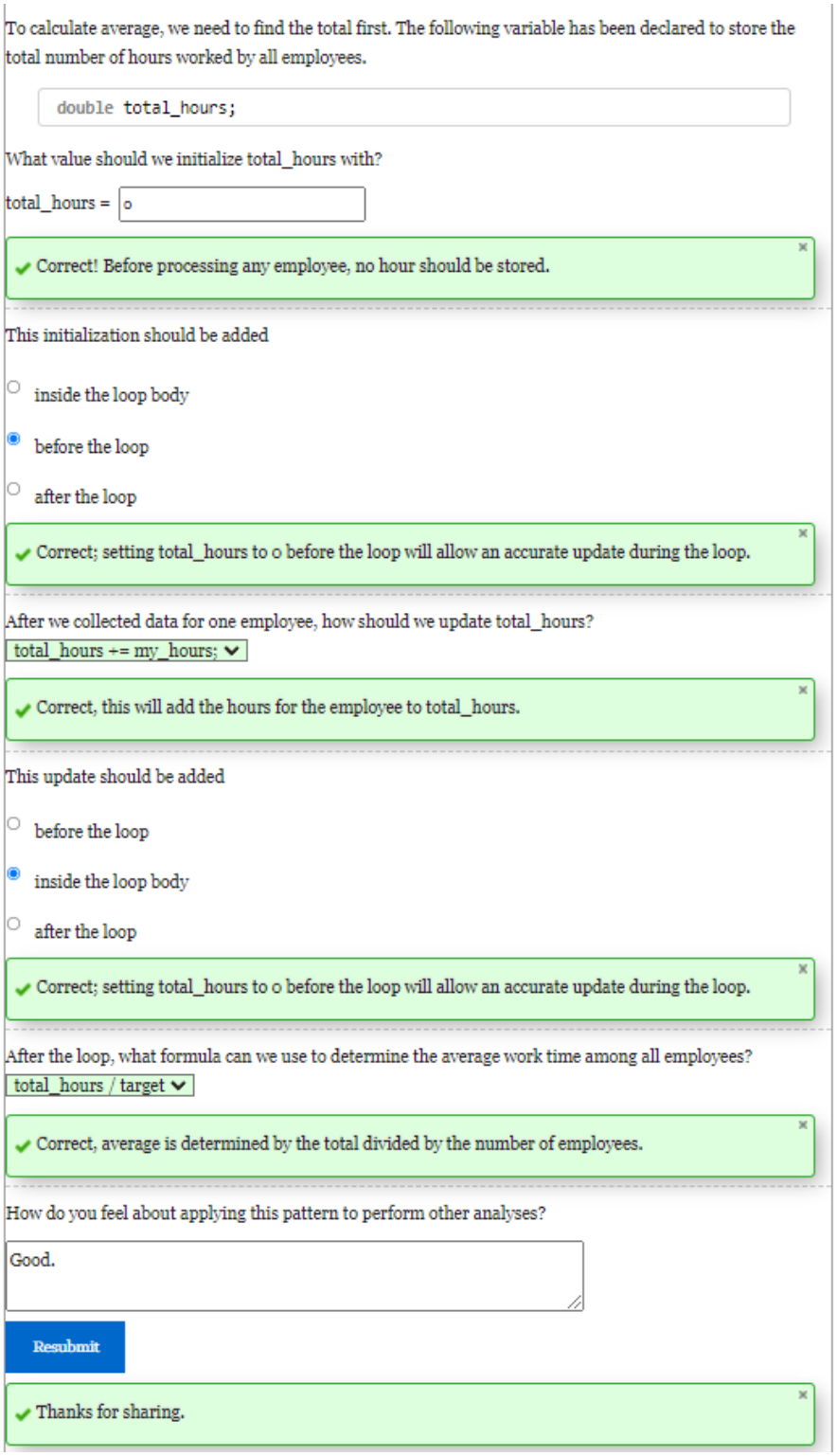
Yes, because of the loop made and calculations.

Resubmit

✔ Thanks for sharing.

Page 30 Loop-based analyses

Hotspot loop pattern for data analyses



Hotspot highest and lowest wage

The above provides a common algorithm pattern in finding the minimum and maximum values. Complete the following to apply the same approach and determine the longest and shortest hours worked by an employee.

```
double longest_hours;
double shortest_hours;
```

Drag the **initial values** to their respective variables **before the loop**. Remember that our program is for weekly wages. There are only 168 hours in 7 days.

199 shortest_hours

-1 longest_hours

✔ Correct, the initial value for `shortest_hours` should be **larger** than all possible hours worked by an employee in a week.

```
if (conditional_expression) {
  longest_hours = my_hours;
}
```

The above conditional statement can be used **inside the loop body** to update `longest_hours` when necessary. Which of the following should we used as the condition expression?

- ○ my_hours < longest_hours
- ◉ longest_hours < my_hours

✔ Correct; this condition indicates the hours worked by the current employee is longer than the longest hours from all **previous** employees.

In the following space, create the conditional statement inside the loop body so that we can update shortest_hours when necessary.

```
If (shortest_hours > my_hours) {
  shortest_hours = my_hours
```

Resubmit

✔

```
if (shortest_hours > my_hours) {
  shortest_hours = my_hours;
}
```

What questions or tips for others do you have regarding the process of finding the minimum and maximum values?

Nothing.

Resubmit

✔ Thanks for sharing.

Page 31 Non-counter based while loops

LBD guess loop condition

The following code segment prompts and collects a guess from the user. It then displays a message to let the user know whether the guess is too big, too small, or just right.

```
1  cout << "What is your guess?t";
2  cin >> guess;
3  if (guess > number)
4    cout << "Sorry, too big";
5  else if (guess < number)
6    cout << "Sorry, too small";
7  else
8    cout << "Great, just right!";
```

When should we prompt the user to provide another guess?

- ○ guess == number
- ◉ guess != number

✔ Correct.

LBD grains loop condition

What should we replace the _loop_condition_? As a reminder, the program will execute the loop body when the loop condition evaluates to true.

- ○ grains_awarded > total_grains
- ◉ grains_awarded < total_grains
- ○ square < target

✔ Correct!

Checkpoint vending matching

Mark all of the statements in the above code segment that should be executed before the loop and do not need to be repeated for each coin.

- [x] int item_cost = 125;
- [x] int total_collected = 0;
- [x] int one_entry;
- [ ] cin >> one_entry;

Check My Answer

✔ Correct, these are the steps that happen only once.

Which of the following must be included in the loop body?

- ○ cin >> one_entry;
- ○ total_collected += one_entry;
- ◉ Both of the above

✔ Correct, we need to collect the coin and update the total.

Which of the following should be used for the loop condition?

- ○ one_entry< item_cost
- ◉ total_collected < item_cost
- ○ one_entry > item_cost
- ○ total_collected > item_cost

✔ Correct, we want to collect another coin when not enough fund is collected.

LBD infinite loop

Which of the following update statement would cause the above loop to be infinite?

- [ ] counter--;
- [x] counter++;

✔ Correct; `counter` starts at 10, `counter++` makes `counter` bigger each iteration and it will never be able to make the condition `counter > 0` false.

```
int counter = 0;
while (counter < 8) {
  cout << "*";
  counter *= 2;
}
```

Will the above loop be an infinite loop?

- [x] Yes
- [ ] No

✔ Correct! `counter` is repeatedly set to 0 and never reaches 8. 0 * 2 results in 0 each time so the loop keeps running!

```
int counter = 0;
while (counter < 8) {
  cout << "*";
  counter += 2;
}
```

Will the above loop be an infinite loop?

- [x] No
- [ ] Yes

✔ Correct! `counter` is incremented by 2 each iteration and will eventually become greater than or equal to 8 resulting in the loop stopping.

LBD no repetition

```
int guess = 100;
int number = rand() % 10 + 1; //a number between 1 and 10
while (_loop_condition_) {
  cin >> guess;
}
```

Which of the following loop condition would have prevented the program from collecting any guesses?

- (•) guess == number
- ( ) guess != number

✔ Correct! the value of number is between 1 and 10. this condition is false when the program reaches the loop. The loop body will not be executed.

```
int countdown = 5;
while (countdown <= 0) {
  cout << "*";
  countdown--;
}
```

Would the above loop display any stars?

- ( ) Yes
- (•) No

✔ Correct! `counter` starts at `5` and `5 <= 0` is `false`. So the loop body does not run.

LBD one-off

Carefully trace the above code segment and predict what it would display:

- ***
- **
- ****

✔ Correct!

```
int counter = 1;
while (counter < 8) {
  cout << "*";
  counter *= 2;
}
```

Carefully trace the above code segment and predict what it would display:

- ****
- ***
- **

✔ Correct! `counter` grows exponentially `3` times before hitting `8`.

```
int counter = 0;
while (counter < 8) {
  cout << "*";
  counter += 2;
}
```

Carefully trace the above code segment and predict what it would display:

- ****
- *****
- ***

✔ Correct, the stars are displayed when counter is 0, 2, 4, and 6.

Page 32 the do … while loop

Checkpoint the guessing game

We’ll work on this in class.

LBD input validation

We would like the user to enter 'Y' or 'N' as the proper response to the question. If they do, we are done with the loop and are ready to move forward. If they don't, we would like to keep them in the loop by asking the same question again.

```
char response;
do {
  cout << "Would you like to play again (Y/N)? ";
  cin >> response;
} while (_loop_condition_);
```

Which of the following would be a good _loop_condition_ to capture invalid user entries and keep asking the user until a valid entry is provided?

- (●) response != 'Y' && response != 'N'
- ( ) response != 'Y' || response != 'N'

✔ Correct.

```
char response;
do {
  cout << "What is your favorite color?\n";
  cout << "A. Red\tB. Green\tC. Blue\tD. Other\n";
  cin >> response;
} while (response >= 'A' && response <= 'D');
```

Which of the following user entry would keep the program in the loop.

- ( ) Z
- (●) A

✔ Correct, this new loop condition would keep the program in the loop for valid entries.

Complete the next few questions based on the following code segment:

```
int age;
do {
  cout << "How old are you?";
  cin >> age;
} while (age < 4 || age > 10);
```

Enter a number for `age` that is too small to break out of the loop. 3

✔ Correct! This number would keep the program in the loop and prompt for another user entry.

Enter a number for `age` that is too large to break out of the loop: 11

✔ Correct! This number would keep the program in the loop and prompt for another user entry.

Enter a number for `age` that would actually break out of the loop: 5

✔ Correct! This is considered a valid age for elementary school children and will break out of the loop.

LBD short-circuit evaluation

For each of the value of response, indicate whether short-circuit evaluation applies.

response	response != 'Y'	Short-circuit evaluation
'Y'	false	Yes
'N'	true	No
'T'	true	No

✔ Correct, because the lhs operand is `false`.

✔ Correct, we still need to evaluate response != 'N'.

✔ Correct, we still need to evaluate response != 'N'.

```
(response == 'Y' || response == 'N')
```

For each of the value of response, indicate whether short-circuit evaluation applies.

response	response == 'Y'	Short-circuit evaluation
'Y'	true	Yes
'D'	false	No
'N'	false	No

✔ Correct, short-circuit evaluation occurs for an `or` operator if its lhs operand is `true`. The fact that `response == 'Y'` is `true` does guarantees the whole expression will be `true`.

✔ Correct, short-circuit evaluation only occurs for an `or` operator if its lhs operand is `true`. The fact that `response == 'Y'` is `false` does not guarantee the whole expression to be either `true` or `false`.

✔ Correct, short-circuit evaluation only occurs for an `or` operator if its lhs operand is `true`. The fact that `response =='Y'` is `false` does not guarantee the whole expression to be either `true` or `false`.

```
(age < 4 || age > 10)
```

Enter a number for age that would cause short circuit evaluation for the above expression:

3

✔ Correct!

```
(age >= 4 && age < 10)
```

Enter a number that would trigger short-circuit evaluation 1

✔ Correct!