Activity #8: For Loops
Recorder's Report

Manager:	Reader:			
Recorder:	Driver:			
Date:	Score:	Satisfactory	/	Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

a) Model 1, Question #5

b) Model 2, Question #13 (write the whole code snippet)

c) Model 3, Question #18

Activity #8: For Loops

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to for loops in C++.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the difference between a `while` loop and a `for` loop.
- Explain the syntax of a `for` loop in C++.
- Explain how an **accumulator** is used in a `for` loop.
- Explain how the **increment operator** and **decrement operator** work.

Process Skill Goals

During the activity, students should make progress toward:

- Write code that includes `for` loops.

Model 1 Two C++ Programs

```
 1  #include <iostream>
 2  #include <string>
 3
 4  using namespace std;
 5
 6  int main() {
 7    string name;
 8    cout << "Enter your name: ";
 9    cin >> name;
10    int x = 0;
11    while (x < 10) {
12      cout << name << endl;
13      x = x + 1;
14    }
15    cout << "Nice to meet you!" << endl;
16  }
17
```

```
 1  #include <iostream>
 2  #include <string>
 3
 4  using namespace std;
 5
 6  int main() {
 7  string name;
 8    cout << "Enter your name: ";
 9    cin >> name;
10    for (int x = 0; x < 10; x += 1) {
11      cout << name << endl;
12    }
13    cout << "Nice to meet you!" << endl;
14  }
15
```

Refer to Model 1 above as your team develops consensus answers to the questions below.

Questions (20 min) Start time:

1. What is the output of each program? Note that you saw the first program in the previous activity and `activity08a.cpp` contains the second program.

2. The loop shown on the right is called a `for` loop. Identify the code that makes up each part of the loop and the line on which it appears.

a) The Initialization Statement:

b) The Test Condition:

c) The Update Statement:

3. You should have noted above that both programs produce the same output. Which is more concise?

4. What output will each of the following code snippets produce? You can use `activity08a.cpp` to check your answers.

a)

```
for (int i = 0; i < 5; i += 1) {
  cout << i << " ";
}
```

b)

```
for (int i = 1; i < 5; i += 1) {
  cout << i << " ";
}
```

c)

```
for (int i = 2; i <= 6; i += 1) {
  cout << i << " ";
}
```

d)

```
for (int i = 2; i <= 6; i += 2) {
  cout << i << " ";
}
```

5. Complete the missing code in the `for` loops below so that they print the indicated output.

a) Even numbers from 100 to 200, inclusive   b) `5 4 3 2 1 0`

```
for (                              ) {
  cout << i << " ";
}
```

```
for (                              ) {
  cout << i << " ";
}
```

6. Based on your solutions above, answer the following questions about the `for` loop.

a) Why do we start a for loop with something like `int i=1` instead of `int i==1`?

b) Do you think we always need the `int` in front of the `i=1` in the `for` loop initialization?

c) Is it better to use a `for` loop when you know how many times the loop should execute (*counter-controlled*) or when you don't know (*sentinel-controlled*)?

7. Rewrite the following `while` loop as a `for` loop that does the same thing.

```
int cnt = 20;
while (cnt >= 10) {
  cnt -= 2;
  cout << cnt << " ";
}

```

Model 2 Another C++ Program

```cpp
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6    int number;
7    int total = 0;
8    for (int i = 0; i < 5; i += 1) {
9      cout << "Enter a number: ";
10     cin >> number;
11     total += number;
12   }
13   cout << "The total is: " << total << endl;
14 }
15
```

Refer to Model 2 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

8. The code for this program is in `activity08b.cpp`. Run it and explain what the program does.

9. Explain what each of the indicated lines of code from the model above does.

a) Line 7:

b) Line 8:

c) Line 11:

10. An *accumulator* is a variable that stores the sum of a group of values. Which variable in this model is an accumulator? Check all that apply.

`number` `total` `i`

11. Why is the variable `total` initialized to zero in line 7 of the model?

12. Would it be possible to use the same variable as both a counter and an accumulator? Explain.

13. An accumulator can also store the product of a set of numbers. How, if at all, would you change the following lines of the model to compute 5! ($5! = 1 \times 2 \times 3 \times 4 \times 5$ is called "five factorial").

a) How would you change `int total = 0;` on line 7?

b) How would you change `for (int i = 0; i < 5; i += 1)` on line 8?

c) How would you change lines 9-10 in the model?

d) How would you change `total += number` on line 11 of the model?

Model 3 Increment / Decrement Operators

Initial x Value	Statement	Final y value	Final x Value
2	`y = x++;`	2	3
2	`y = ++x;`	3	3
2	`y = x--;`	2	1
2	`y = --x;`	1	1

Refer to Model 3 above as your team develops consensus answers to the questions below.

Questions (15 min) Start time:

14. Based on the model above, describe what each operator does.

a) `++`

b) `--`

15. Rewrite the following `for` loop to use an *increment operator*.

```
for (int i=5; i<=10; i+=1) {
  cout << i << " ";
}

```

16. Rewrite the following `while` loop to use an *decrement operator*.

```
int counter = 10;
while (counter > 0) {
  cout << "text" << endl;
  counter = counter - 1;
}

```

17. Why are increment and decrement operators especially useful for counters?

18. Note that there are two versions of the *increment operator*: a *pre-increment* version (i.e. `++x`) and a *post-increment* version (i.e. `x++`). The same is true of the decrement operator. Use the file `activity08c.cpp` to assist as you complete the missing entries in the table below.

Initial x Value	Expression	Expression Value	Final x Value
5	`3 + 2*(x++)`		6
3	`5 - (++x) / 2`	3	
	`6 * (--x)`	0	0
-3		-1	-4

19. Based on your work above, what is the difference between the pre- and post- versions of these operators?