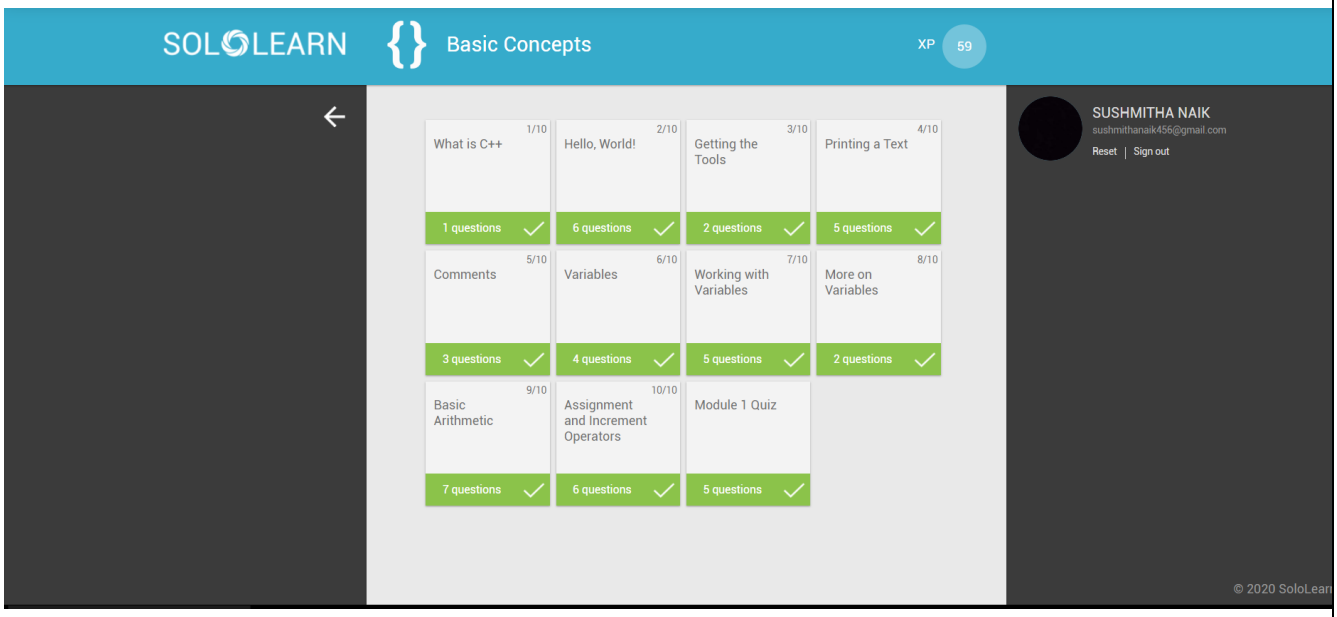
DAILY ASSESSMENT FORMAT

Date:	**22 June 2020**	**Name:**	**Sushmitha R Naik**
Course:	**C++ (Sololearn)**	**USN:**	**4al17ec090**
Topic:	**Module 1: Basic Concepts**	**Semester & Section:**	**6th b**
GitHub Repository:	**Sushmitha_naik**		

FORENOON SESSION DETAILS

Image of session



Report:

C++:

C++ is a general-purpose programming language.

C++ is used to create computer programs. Anything from art applications, music players and even video games

A C++ program is a collection of commands or statements.

Below is a simple code that has "Hello world!" as its output.

```
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello world!";
  return 0;
}
```

Main

Program execution begins with the main function, int main().

```
#include <iostream>
using namespace std;

int main()
{
  cout << "Hello world!";
  return 0;
}
```

Getting the Tools

1. Integrated Development Environment (IDE): Provides tools for writing source code. Any text editor can be used as an IDE.

2. Compiler: Compiles source code into the final executable program. There are a number of C++ compilers available. The most frequently used and free available compiler is the GNU C/C++ compiler.

New Line

The cout operator does not insert a line break at the end of the output.

One way to print two lines is to use the endl manipulator, which will put in a line break.

```
#include <iostream>
using namespace std;

int main()
{
cout << "Hello world!" << endl;
cout << "I love programming!";
return 0;
}
```

Comments:

Comments are explanatory statements that you can include in the C++ code to explain what the

code is doing.
The compiler ignores everything that appears in the comment, so none of that information shows in the result.

A comment beginning with two slashes (//) is called a single-line comment. The slashes tell the compiler to ignore everything that follows, until the end of the line.

For example:

```
#include <iostream>
using namespace std;

int main()
{
// prints "Hello world"
cout << "Hello world!";
return 0;
}
```

Variables:

- **Creating a variable reserves a memory location, or a space in memory for storing values. The compiler requires that you provide a data type for each variable you declare.**
- **C++ offer a rich assortment of built-in as well as user defined data types.**

  Integer, a built-in type, represents a whole number value. Define integer using the keyword int.
  C++ requires that you specify the type and the identifier for each variable defined.
  An identifier is a name for a variable, function, class, module, or any other user-defined item.
- **An identifier starts with a letter (A-Z or a-z) or an underscore (_), followed by additional letters, underscores, and digits (0 to 9).**
  For example, define a variable called myVariable that can hold integer values as follows:int myVariable = 10;

Arithmetic Operators

C++ supports these arithmetic operators.

Operator	Symbol	Form
Addition	+	x + y
Subtraction	-	x - y
Multiplication	*	x * y
Division	/	x / y
Modulus	%	x % y

Assignment Operators:

- **The simple assignment operator (=) assigns the right side to the left side.**
  C++ provides shorthand operators that have the capability of performing an operation and an assignment at the same time.
  For example:int x = 10;
  x += 4; // equivalent to x = x + 4
  x -= 5; // equivalent to x = x – 5

- **Increment Operator**

- **The increment operator is used to increase an integer's value by one, and is a commonly used C++ operator.**
- **x++; //equivalent to x = x + 1**

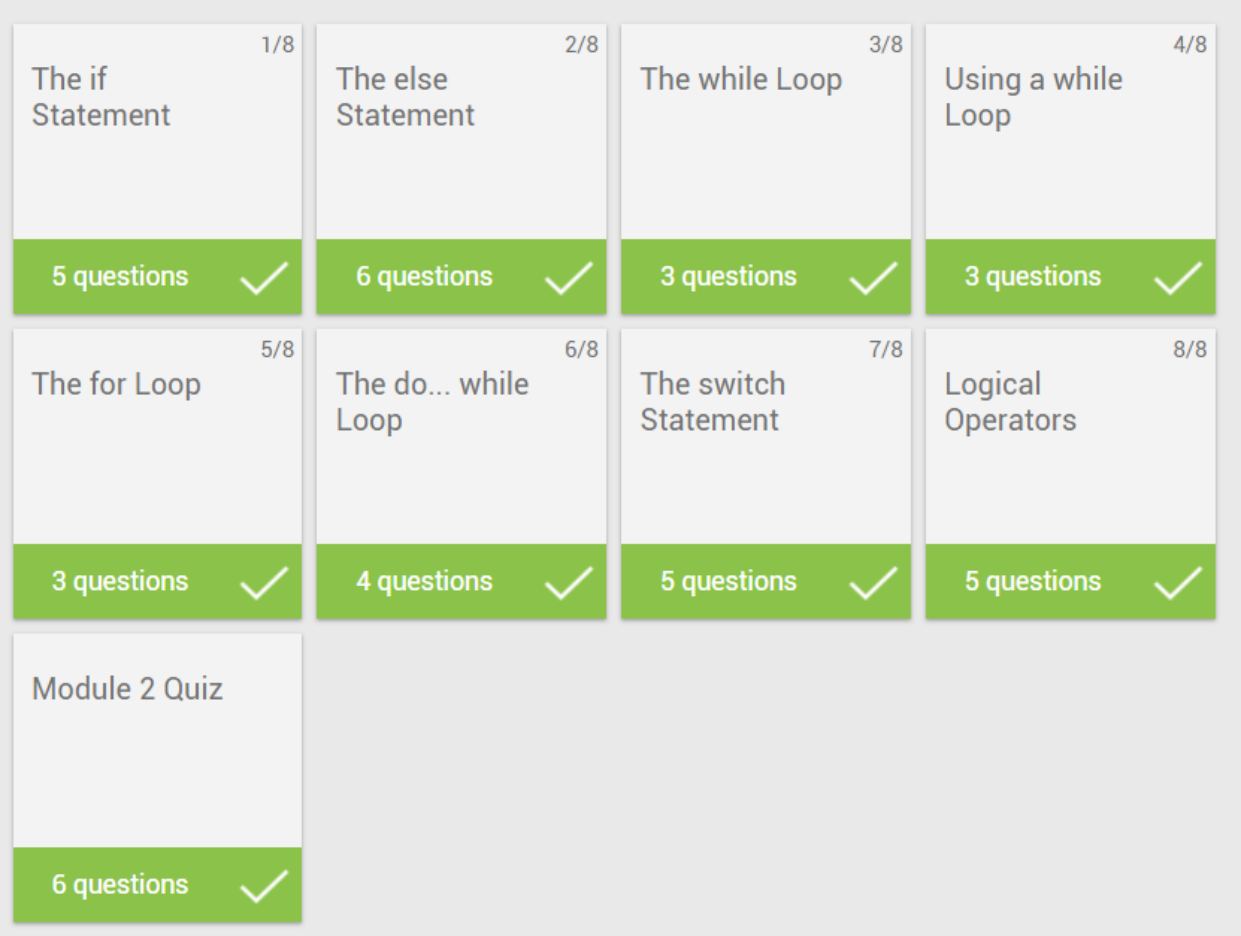
- **The increment operator has two forms, prefix and postfix.**
- **++x; // prefix**
- **x++; // postfix**

- **Prefix increments the value, and then proceeds with the expression.**
- **Postfix evaluates the expression and then performs the incrementing.**

Date:	22 June 2020	Name:	Sushmitha R Naik
Course:	C++ (Sololearn)	USN:	4al17ec090
Topic:	Module 2: Conditionals and Loops	Semester&Section :	6th b
Git hub repository	Sushmitha_naik		

AFTERNOON SESSION DETAILS

Image of session



Report:

Conditionals and Loops

Decision Making

The if statement is used to execute some code if a condition is true.

Syntax:

```
if (condition) {
  statements
}
```

The condition specifies which expression is to be evaluated. If the condition is true, the statements in the curly brackets are executed.

The if Statement

Use relational operators to evaluate conditions.

For example:

```
if (7 > 4) {
 cout <<  "Yes";
}

// Outputs "Yes"
```

The else Statement

An if statement can be followed by an optional else statement, which executes when the condition is false.

Syntax:

```
if (condition) {
  //statements
}
else {
 //statements
}
```

The code above will test the condition:

- If it evaluates to true, then the code inside the if statement will be executed.

- If it evaluates to false, then the code inside the else statement will be executed.

The while Loop:

The loop's body is the block of statements within curly braces.
For example:

```
int num = 1;
while (num < 6) {
  cout << "Number: " << num << endl;
  num = num + 1;
}

/* Outputs
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
*/
```

for loop:

A for loop is a repetition control structure that allows you to efficiently write a loop that executes a specific number of times.
Syntax:

```
for ( init; condition; increment ) {
  statement(s);
}
```

The init step is executed first, and does not repeat.

The do...while Loop

Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop checks its condition at the bottom of the loop.
A do...while loop is similar to a while loop. The one difference is that the do...while loop is guaranteed to execute at least one time.
Syntax:

```
do {
   statement(s);
} while (condition);
```

The switch Statement

The switch statement tests a variable against a list of values, which are called cases, to determine whether it is equal to any of them.

```
switch (expression) {
  case value1:
    statement(s);
    break;
  case value2:
    statement(s);
    break;
    ...
  case valueN:
    statement(s);
    break;
}
```

Logical Operators

Use logical operators to combine conditional statements and return true or false.

Operator	Name of Operator	Form
&&	**AND** Operator	y && y
||	**OR** Operator	x || y
!	**NOT** Operator	! x

The AND operator works the following way: