

Name- Khushi Nitinkumar Patel

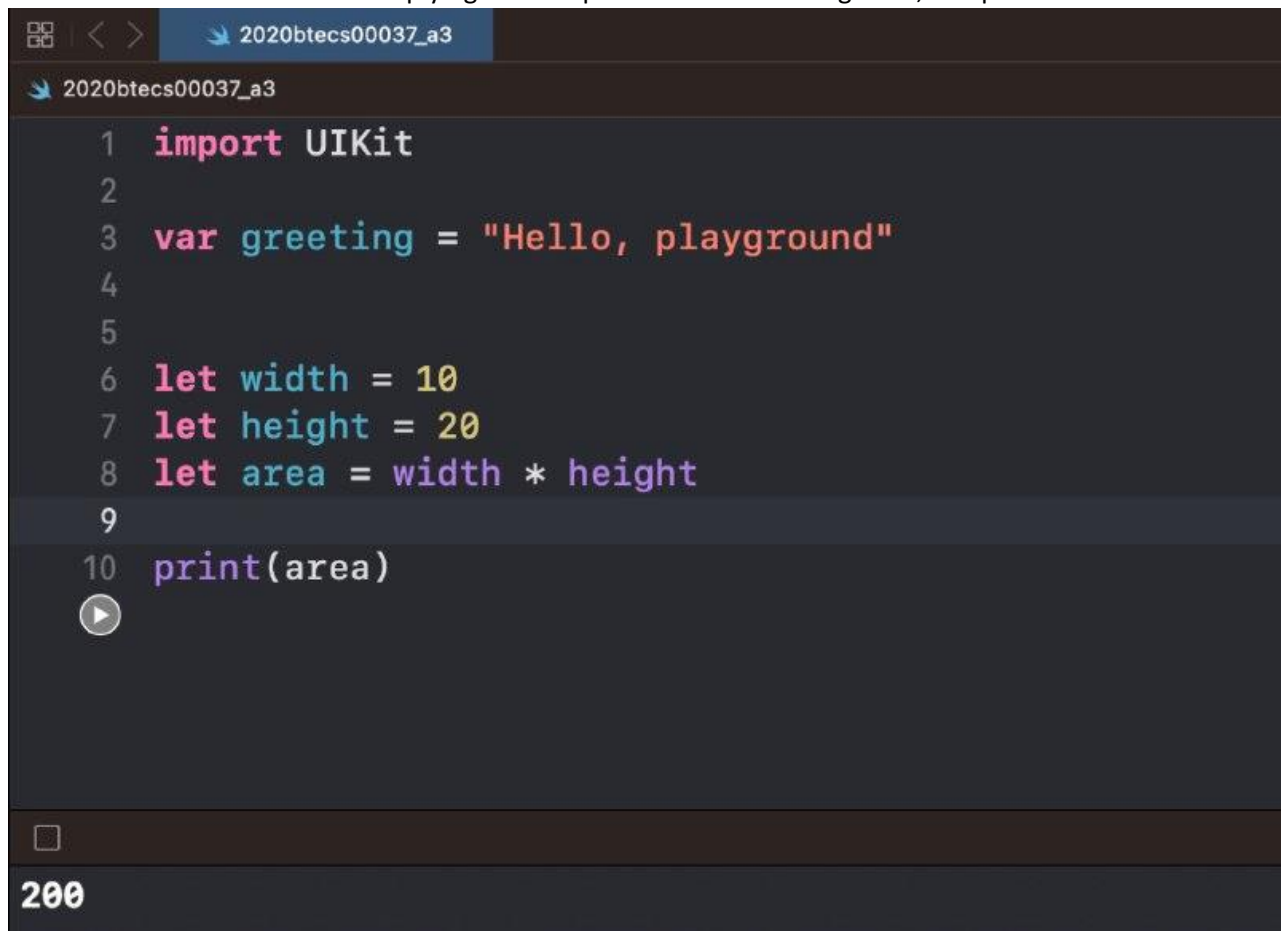
PRN-2020BTECS00037

Batch- T2

Assignment no – 3

Exercise - Basic Arithmetic

1. You decide to build a shed and want to know beforehand the area of your yard that it will take up. Create two constants, `width` and `height`, with values of 10 and 20, respectively. Create an `area` constant that is the result of multiplying the two previous constants together, and print out the result.

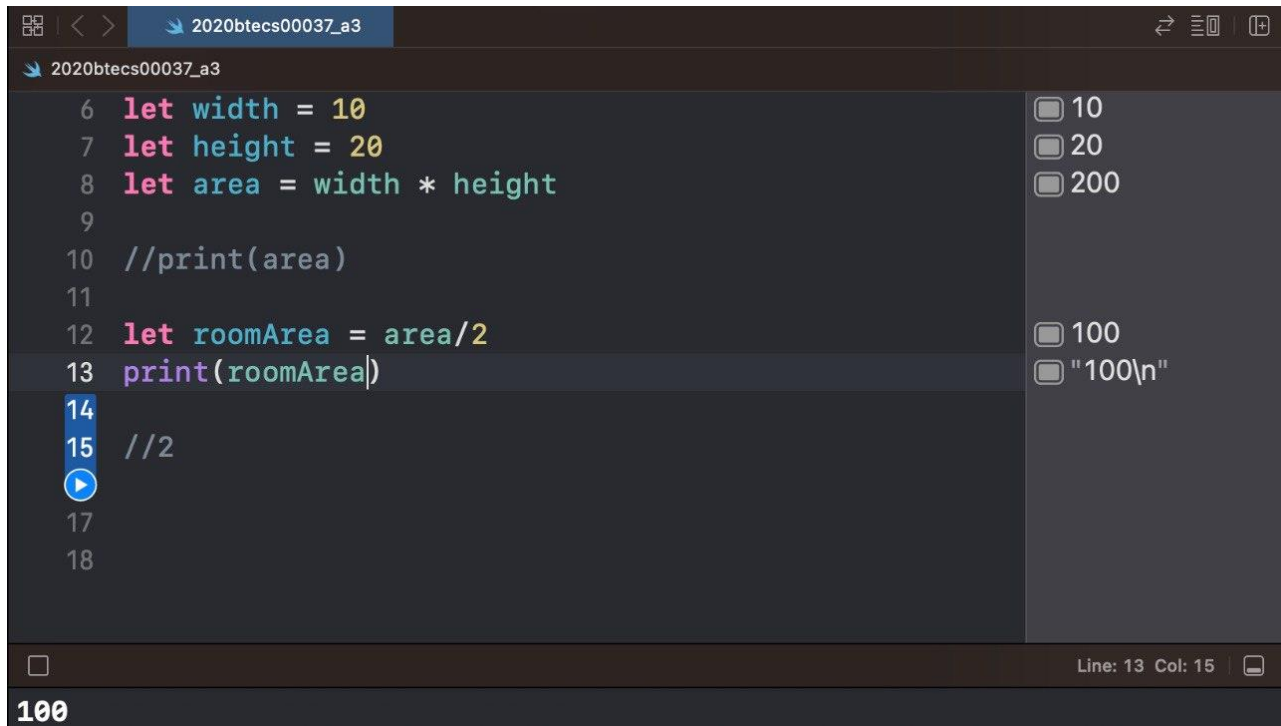


The image shows a Swift Playground window titled "2020btecs00037_a3". The code is as follows:

```
1 import UIKit
2
3 var greeting = "Hello, playground"
4
5
6 let width = 10
7 let height = 20
8 let area = width * height
9
10 print(area)
```

Below the code editor, there is a play button icon. At the bottom of the playground, the output "200" is displayed.

2. You decide that you'll divide your shed into two rooms. You want to know if dividing it equally will leave enough room for some of your larger storage items. Create a `roomArea` constant that is the result of dividing `area` in half. Print out the result.

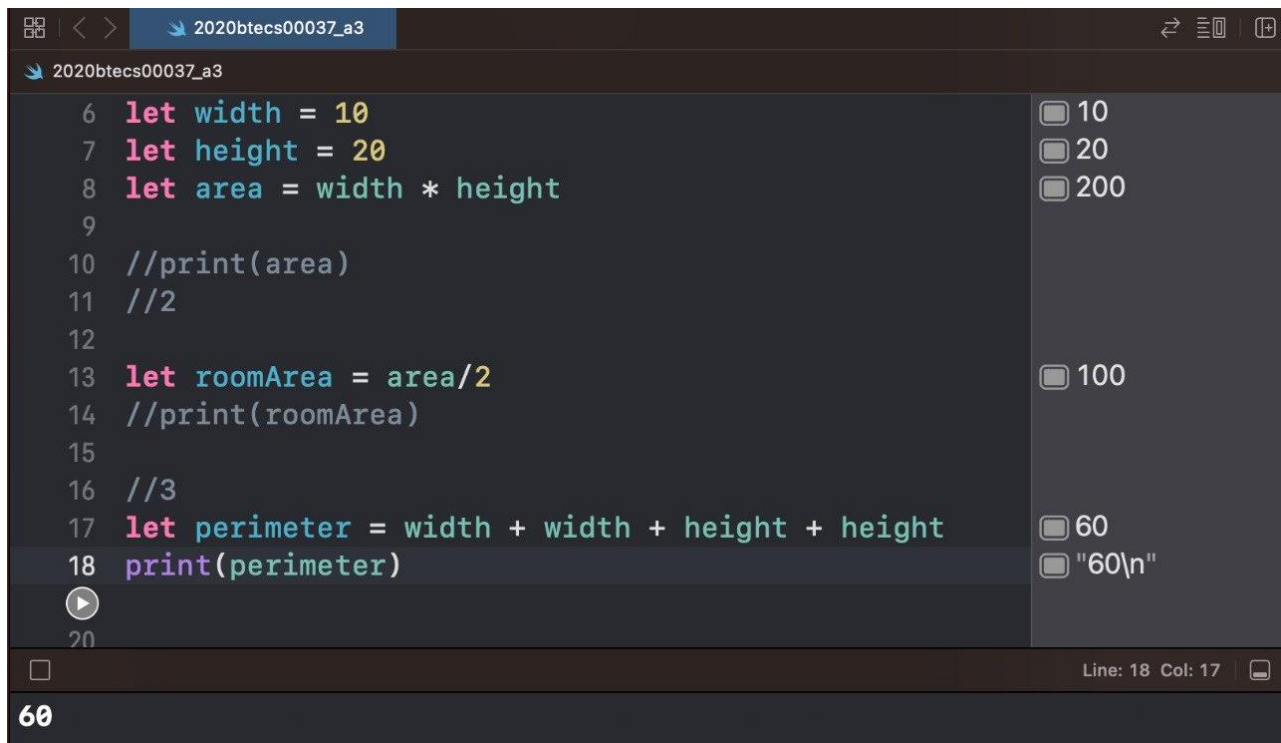


The screenshot shows a code editor with a dark theme. The file name is '2020btcs00037_a3'. The code is as follows:

```
6 let width = 10
7 let height = 20
8 let area = width * height
9
10 //print(area)
11
12 let roomArea = area/2
13 print(roomArea)
14
15 //2
16
17
18
```

On the right side, there are several checkboxes for testing values: 10, 20, 200, 100, and "100\n". The checkbox for 100 is selected. At the bottom, the output shows '100'.

3. Create a `perimeter` constant whose value equals `width` plus `width` plus `height` plus `height`, then print out the result.



The screenshot shows the same code editor with the following code:

```
6 let width = 10
7 let height = 20
8 let area = width * height
9
10 //print(area)
11 //2
12
13 let roomArea = area/2
14 //print(roomArea)
15
16 //3
17 let perimeter = width + width + height + height
18 print(perimeter)
19
20
```

On the right side, there are checkboxes for testing values: 10, 20, 200, 100, 60, and "60\n". The checkbox for 60 is selected. At the bottom, the output shows '60'.

4. Print what you would expect the result of integer division of 10 divided by 3 to be. Create a constant, `integerDivisionResult` that is the result of 10 divided by 3, and print the value.

```
18 //print(perimeter)
19
20 //4
21 let integerDivisionResult = 10/3
22 print("3.33")
23 print(integerDivisionResult)
```

☐ 3
☐ "3.33\n"
☐ "3\n"

Line: 23 Col: 28

3.33
3

5. Now create two constants, `double10` and `double3`, set to 10 and 3, and declare their types as `Double` values. Declare a final constant `divisionResult` equal to the result of `double10` divided by `double3`. Print the value of `divisionResult`. How does this differ from the value when using integer division?

```
24
25 //5
26 let double10 = 10
27 let double3 = 3
28 let divisionResult = double10 / double3
29 print(divisionResult)
```

☐

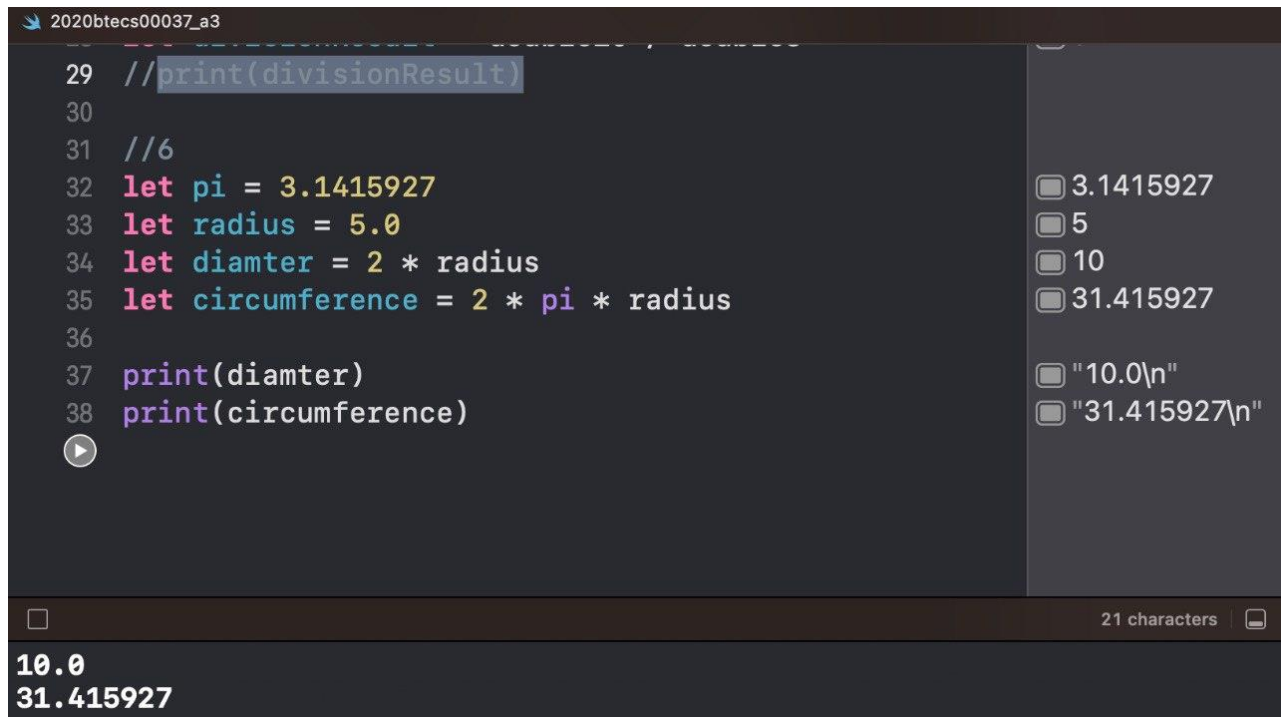
3

6. Given the value pi (3.1415927), create a `radius` constant with a value of 5.0, then calculate the diameter and circumference of the circle using the following equations, and print the results:

```
//diameter = 2 * radius*
```

```
//circumference = 2 * pi * radius.*
```

```
//let pi = 3.1415927
```



The screenshot shows a code editor with the following JavaScript code:

```
29 //print(divisionResult)
30
31 //6
32 let pi = 3.1415927
33 let radius = 5.0
34 let diameter = 2 * radius
35 let circumference = 2 * pi * radius
36
37 print(diameter)
38 print(circumference)
```

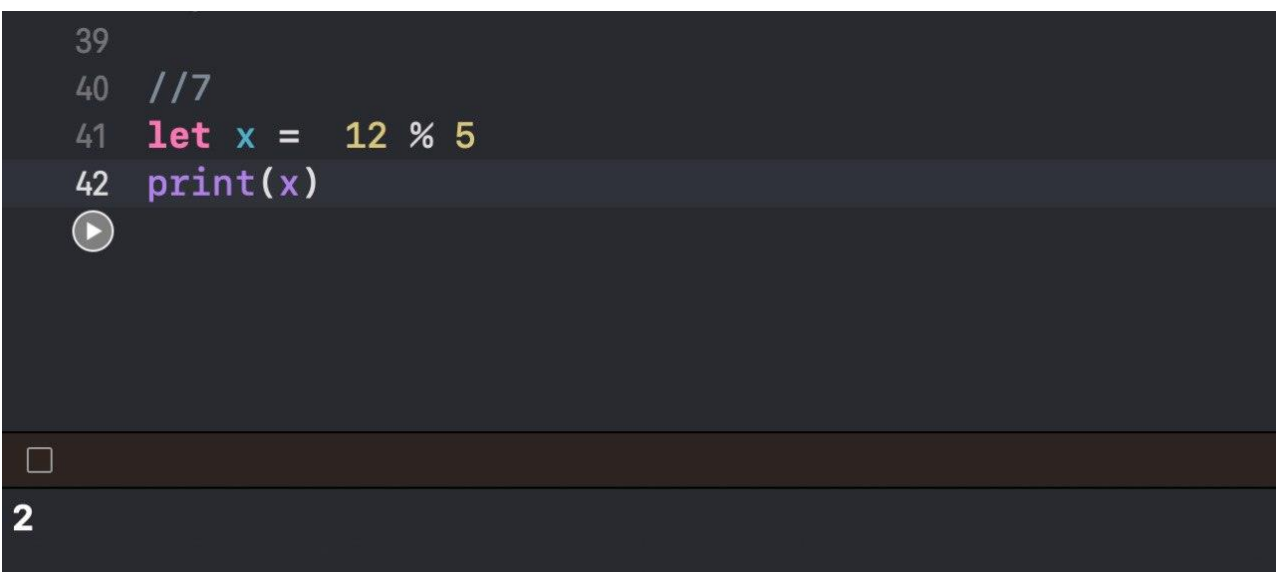
On the right side, there is a list of variables and their values:

- ☐ 3.1415927
- ☐ 5
- ☐ 10
- ☐ 31.415927
- ☐ "10.0\n"
- ☐ "31.415927\n"

At the bottom, the output is displayed:

```
10.0
31.415927
```

7. Create an integer constant. Using the modulus operator, set its value to the remainder of 12 divided by 5.



The screenshot shows a code editor with the following JavaScript code:

```
39
40 //7
41 let x = 12 % 5
42 print(x)
```

At the bottom, the output is displayed:

```
2
```

8. Create two integer constants, `even` and `odd` and set them to any even integer and any odd integer, respectively. For each, print the remainder of dividing the value by 2. Looking at the results, how do you think you could use the remainder operator to determine if an integer is even or odd?

```
44 //8
45
46 let even = 4
47 let odd = 5
48 let remainder_even = even % 2
49 let remainder_odd = odd % 2
50
51 print(remainder_even)
52 print(remainder_odd)
```

☐

0
1

App Exercise - Fitness Calculations

9. Your fitness tracker keeps track of users' heart rate, but you might also want to display their average heart rate over the last hour. Create three constants, `heartRate1`, `heartRate2`, and `heartRate3`. Give each constant a different value between 60 and 100. Create a constant `addedHR` equal to the sum of all three heart rates. Now create a constant called `averageHR` that equals `addedHR` divided by 3 to get the average. Print the result.

```
54 //9
55
56 let heartRate1 = 68
57 let heartRate2 = 74
58 let heartRate3 = 95
59 let addedHR = heartRate1 + heartRate2 + heartRate3
60 let averageHR = addedHR/3
61
62 print(averageHR)
```

☐

79

10. Now create three more constants, `heartRate1D`, `heartRate2D`, and `heartRate3D`, equal to the same values as `heartRate1`, `heartRate2`, and `heartRate3`. These new constants should be of type `Double`. Create a constant `addedHRD` equal to the sum of all three heart rates. Create a constant called `averageHRD` that equals the `addedHRD` divided by 3 to get the average of your new heart rate constants. Print the result. Does this differ from your previous average? Why or why not?

```
66 let heartRate1D = Double(68)
67 let heartRate2D = Double(74)
68 let heartRate3D = Double(95)
69 let addedHR = heartRate1D + heartRate2D +
    heartRate3D
70 let averageHR = addedHR/3
71 print(averageHR)
```



73



79.0

11. Imagine that partway through the day a user has taken 3,467 steps out of the 10,000 step goal. Create constants `steps` and `goal`. Both will need to be of type `Double` so that you can perform accurate calculations. `steps` should be assigned the value 3,467, and `goal` should be assigned 10,000. Create a constant `percentOfGoal` that equals an expression that evaluates to the percent of the goal that has been achieved so far.

```
75 let step = Double(3467)
76 let goal = Double(10000)
77 let percentOfGoal = (step / goal) * 100
78 print(percentOfGoal)
```



80



34.67

Exercise - Compound Assignment

12. Declare a variable whose value begins at 10. Using addition, update the value to 15 using the compound assignment operator. Using multiplication, update the value to 30 using compound assignment. Print out the variable's value after each assignment.

```
82 var z = 10
83 z+=5
84 print(z)
85 z*=2
86 print(z)
87
88
```



15
30

13. Create a variable called `piggyBank` that begins at 0. You will use this to keep track of money you earn and spend. For each point below, use the right compound assignment operator to update the balance in your piggy bank.

- Your neighbor gives you 10 dollars for mowing her lawn
- You earn 20 more dollars throughout the week doing odd jobs
- You spend half your money on dinner and a movie
- You triple what's left in your piggy bank by washing windows
- You spend 3 dollars at a convenience store

Print the balance of your piggy bank after each step.

```
89
90 var piggyBank = 0
91 piggyBank+=10
92 print(piggyBank)
93 piggyBank*=2
94 print(piggyBank)
95 piggyBank/=2
96 print(piggyBank)
97 piggyBank*=3
98 print(piggyBank)
99 piggyBank-=3
100 print(piggyBank)
101
102
```

☐ 0
☐ 10
☐ "10\n"
☐ 20
☐ "20\n"
☐ 10
☐ "10\n"
☐ 30
☐ "30\n"
☐ 27
☐ "27\n"

Line: 101 Col: 1

10
20
10
30
27

App Exercise - Counting

14. The most basic feature of your fitness tracking app is counting steps. Create a variable `steps` and set it equal to 0. Then increment its value by 1 to simulate a user taking a step.

```
102 //14
103
104 var steps = 0
105 steps+=1
106 print(steps)|
▶
108
109
```

☐

1

15. In addition to tracking steps, your fitness tracking app tracks distance traveled. Create a variable `distance` of type `Double` and set it equal to 50. This will represent the user having traveled 50 feet. You decide, however, to display the distance in meters. 1 meter is approximately equal to 3 feet. Use a compound assignment operator to convert `distance` to meters. Print the result.

```
109
110 var distance = Double(50)
111
112
113 distance = distance/3|
114 print(distance)
▶
116
```


☐

16.666666666666668

Exercise - Order of Operations

16. Print out what you think $10 + 2 * 5$ evaluates to. Then print out the actual expression (i.e. ``print(10 + 2 * 5)``)

```
117
118 print(30)
119 print(10 + 2 * 5)
```




☐

30
20

17. In a separate ``print`` statement, add in the necessary parentheses so that addition takes place before multiplication. Print out what you think $4 * 9 - 6 / 2$ evaluates to. Then print out the actual expression.

```
120
121 //17
122 print(33)
123 print((10+2)*5)
124 print(4 * 9 - 6/2)
```



☐

33
60
33

18. In a separate `print` statement, add in the necessary parentheses so that the subtraction is prioritized over the multiplication and division.

```
126 //18
127 print(4 * (9 - 6)/2)
129
```

☐

6

App Exercise - Complex Fitness Calculations

19. If you completed the Fitness Calculations exercise, you calculated an average heart rate to display to the user. However, using proper order of operations you can do this in fewer steps. Create three separate heart rate constants, all of type `Double`, with values between 60 and 100. Then create a constant equal to the average heart rate. If you use correct order of operations you can do the heart calculation in one line.

```
130
131 let hr1 = Double(68)
132 let hr2 = Double(77)
133 let hr3 = Double(85)
134 let avgHeartRate = (hr1 + hr2 + hr3)/3
135 print(avgHeartRate)
137
138
```

☐

76.66666666666667

20. One feature you might want to give users is to display their current body temperature. Create a constant `tempInFahrenheit` equal to 98.6. You may want to also show the temperature in celsius. You can convert fahrenheit to celsius by taking `tempInFahrenheit` and subtracting 32, then multiplying the result by (5.0/9.0). Create a constant `tempInCelsius` that calculates in one line the temperature in celsius.

```
137 //20
138 let tempInFahrenheit = 98.6
139 let tempInCelsius = (tempInFahrenheit - 32) *
    (5.0/9.0)
140 print(tempInCelsius)
```



37.0

Exercise - Numeric Type Conversion

21. Create an integer constant `x` with a value of 10, and a double constant `y` with a value of 3.2. Create a constant `multipliedAsIntegers` equal to `x` times `y`. Does this compile? If not, fix it by converting your `Double` to an `Int` in the mathematical expression. Print the result.

```
142 //21
143 let x = 10
144 let y = Double(3.2)
145 let multipliedAsIntegers = x * Int(y)
146
147 print(multipliedAsIntegers)
```

30

22. Create a constant `multipliedAsDoubles` equal to `x` times `y`, but this time convert the `Int` to a `Double` in the expression. Print the result.

```
117
118 let multipliedAsDoubles = Double(x) * y
119 print(multipliedAsDoubles)
120
121
```

32.0

23. Are the values of `multipliedAsIntegers` and `multipliedAsDoubles` different? Print a statement to the console explaining why.

```
120
121 print("Int(y) will be converted from 3.2 to 3 and hence multipliedAsIntegers will be 10 * 3 = 30")
122 print("Double(x) will be converted from 10 to 10.0 and hence multipliedAsDoubles will be 10.0 * 3.2 = 32.0")
123
124
125
```

Int(y) will be converted from 3.2 to 3 and hence multipliedAsIntegers will be 10 * 3 = 30
Double(x) will be converted from 10 to 10.0 and hence multipliedAsDoubles will be 10.0 * 3.2 = 32.0

App Exercise - Converting Types

24. If you completed the Fitness Calculations exercise, you calculated the percent of the daily step goal that a user has achieved. However, you did this by having `steps` be of type `Double`. But you can't really track a partial step, so `steps` should probably be of type `Int`. Go ahead and declare `steps` as type `Int` and give it a value between 500 and 6,000. Then declare `goal` as type `Int` and set it equal to 10,000.

```
157
158 var steps:Int
159 steps = 9000|
160 var goal:Int
161 goal = 10_000
162 let percentOfGoal:Double
163 percentOfGoal = Double(steps*100)/10_000
164 print(percentOfGoal,"%")
165
```

90.0 %

25. Now create a constant `percentOfGoal` of type `Double` that equals the percent of the goal that has been reached so far. You'll need to convert your constants of type `Int` to be of type `Double` in your calculation.

```
157
158 var steps:Int
159 steps = 9000|
160 var goal:Int
161 goal = 10_000
162 let percentOfGoal:Double
163 percentOfGoal = Double(steps*100)/10_000
164 print(percentOfGoal,"%")
165
```

90.0 %
