1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. cd: change directory
    b. ls: list file in the current directory
    c. pwd: print working directory

2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	**Suggested Data Type**
A person's name	string
A person's age in years	integer
A phone number	string or integer
A temperature in Celsius	float
The average age of a group of people	integer
Whether a person has eaten lunch	boolean

3. Aside from the examples already provided in question 2, come up with an example of information that could be stored as:

Data type	**Suggested Information**
String	Person name
Integer	Product quantity
Float	Height
Boolean	Did I sleep?

4. Fill out the last two columns of the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	**Given**	**Value**	**Data Type**
`6`		6	Integer
`True`		True	Boolean

`a`	`a = 2.5`	2.5	Integer
`1 + 2 * 3`		7	Integer
`a and False`	`a = True`	True	Boolean
`a or False`	`a = True`	True	Boolean

`a + b`	`a = 1`
	`b = 2`
		3	Integer

`2 * a`	`a = 3`	6	Integer

`a * 2 + b`	`a = 2.5`
	`b = 2`
		7.0	Integer

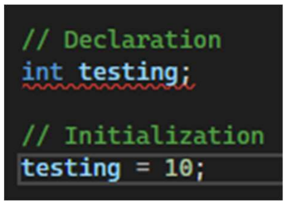
`a + 2 * b`	`a = 2.5`
	`b = 2`
		6.5	Float

`(a + b) * c`	`a = 1`
	`b = 1`
	`c = 5`
		10	Integer

`"Fred" + " Smith"`		Fred Smith	String
`a + " Smith"`	`a = "Wilma"`	Wilma Smith	string

5. Using an example, explain the difference between **declaring** and **initialising** a variable.

The difference between the two is declaring specifies the variable type and name while initialising assign a value to the declared variable

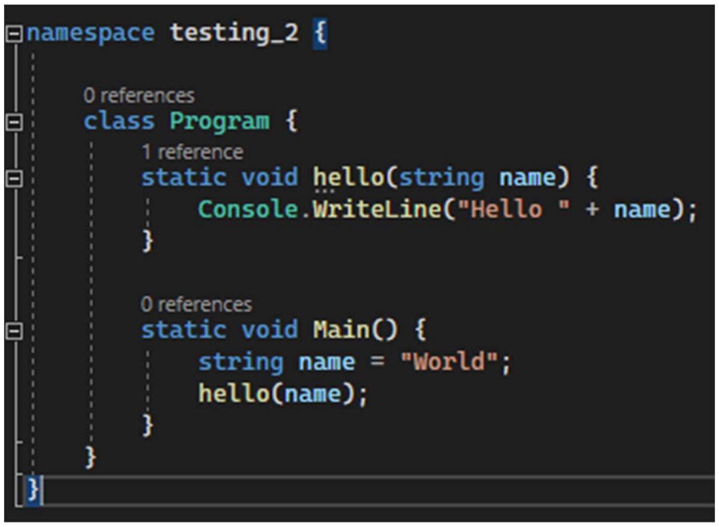


```
// Declaration
int testing;

// Initialization
testing = 10;
```

6. Explain the term **parameter**. Write some code that demonstrates a simple of use of a parameter. You should show a procedure or function that uses a parameter, and how you would call that procedure or function.

A parameter is the variable or value that is passed to the function as input



```
namespace testing_2 {

    0 references
    class Program {
        1 reference
        static void hello(string name) {
            Console.WriteLine("Hello " + name);
        }

        0 references
        static void Main() {
            string name = "World";
            hello(name);
        }
    }
}
```

7. Using an example, describe the term **scope** as it is used in procedural programming (not in business or project management). Make sure you explain the different kinds of scope.

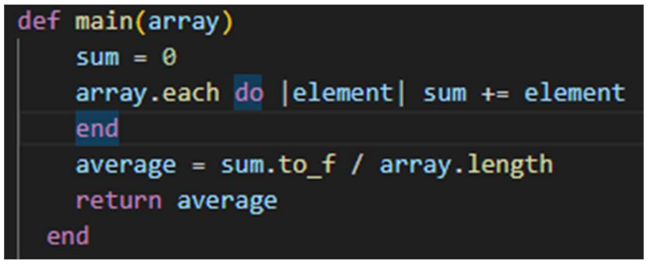
Scope is where a variable can be accessed and refferenced in the program

Local Scope: Can only be access in the function or nested function that it is declared in

Global Scope: can be access anywhere in the program

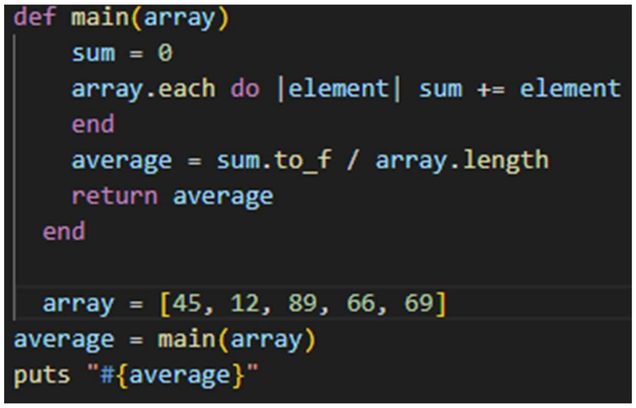
8. In a procedural style, in any language you like, write a function called Average, which accepts an array of integers and returns the average of those integers. Do not use any libraries for calculating the average. You must demonstrate appropriate use of parameters, returning and assigning values, and use of a loop. Note — just write the function at this point, we'll *use* it in the next task. You shouldn't have a complete program or even code that outputs anything yet at the end of this question.

I did this in Ruby:



```
def main(array)
    sum = 0
    array.each do |element| sum += element
    end
    average = sum.to_f / array.length
    return average
  end
```

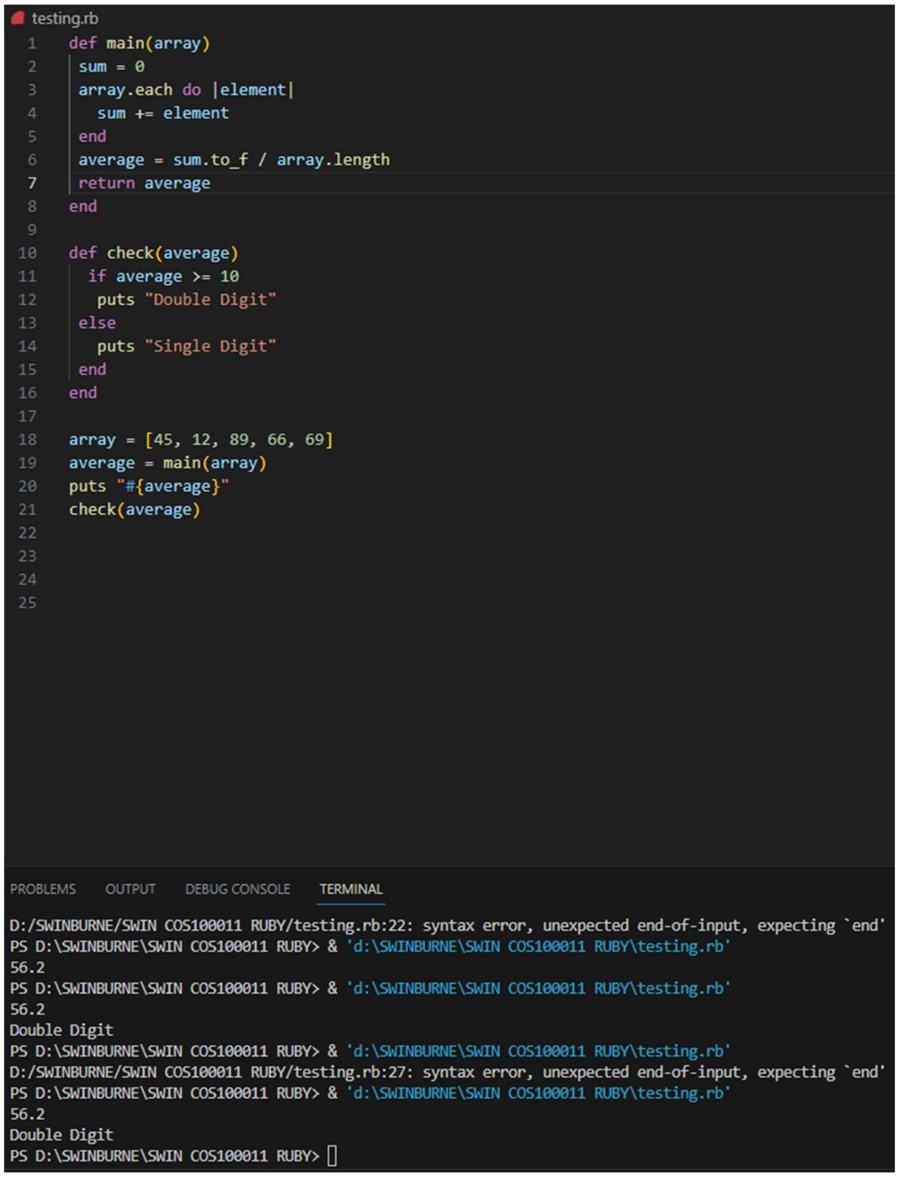
9. In the same language, write the code you would need to call that function and print out the result.



```ruby
def main(array)
    sum = 0
    array.each do |element| sum += element
    end
    average = sum.to_f / array.length
    return average
  end

  array = [45, 12, 89, 66, 69]
average = main(array)
puts "#{average}"
```

10. To the code from 9, add code to print the message “Double digits” if the average is above or equal to 10. Otherwise, print the message “Single digits”. Provide a screenshot of your program running.



testing.rb

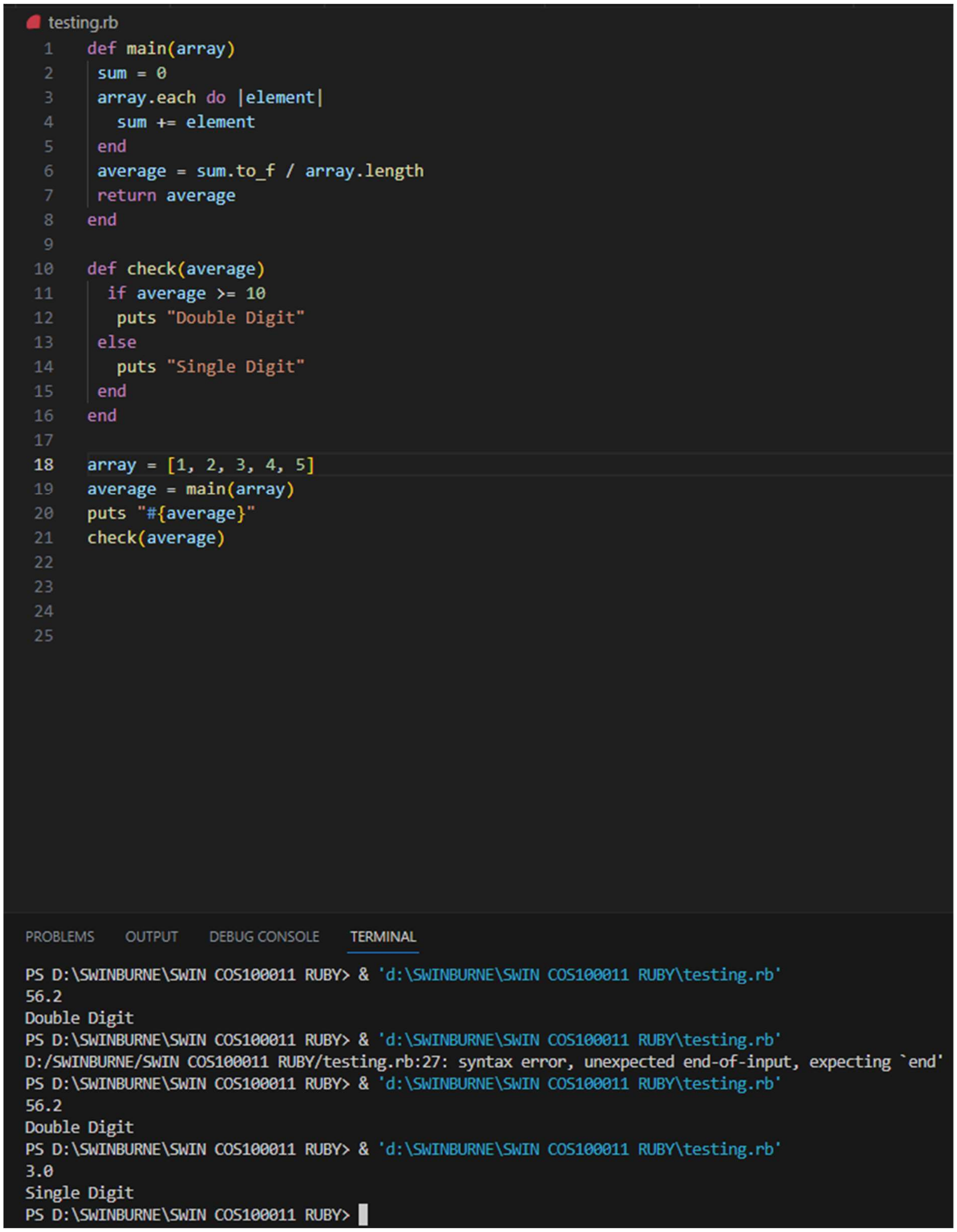
```ruby
def main(array)
 sum = 0
 array.each do |element|
   sum += element
 end
 average = sum.to_f / array.length
 return average
end

def check(average)
  if average >= 10
   puts "Double Digit"
 else
   puts "Single Digit"
 end
end

array = [45, 12, 89, 66, 69]
average = main(array)
puts "#{average}"
check(average)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:/SWINBURNE/SWIN COS100011 RUBY/testing.rb:22: syntax error, unexpected end-of-input, expecting `end'
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
56.2
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
56.2
Double Digit
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
D:/SWINBURNE/SWIN COS100011 RUBY/testing.rb:27: syntax error, unexpected end-of-input, expecting `end'
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
56.2
Double Digit
PS D:\SWINBURNE\SWIN COS100011 RUBY> 
```



testing.rb

```ruby
def main(array)
  sum = 0
  array.each do |element|
    sum += element
  end
  average = sum.to_f / array.length
  return average
end

def check(average)
   if average >= 10
    puts "Double Digit"
  else
    puts "Single Digit"
  end
end

array = [1, 2, 3, 4, 5]
average = main(array)
puts "#{average}"
check(average)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
56.2
Double Digit
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
D:/SWINBURNE/SWIN COS100011 RUBY/testing.rb:27: syntax error, unexpected end-of-input, expecting `end'
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
56.2
Double Digit
PS D:\SWINBURNE\SWIN COS100011 RUBY> & 'd:\SWINBURNE\SWIN COS100011 RUBY\testing.rb'
3.0
Single Digit
PS D:\SWINBURNE\SWIN COS100011 RUBY>
```