

T1A1 - Workbook

Q01

- **Usenet** Starting out as a collection of Bourne Shell scripts (The precursor to BASH) by two university students, “Tom Truscott” & “Jim Ellis”. And going public in 1980. Usenet is the earliest example of what a modern internet user may recognize as the internet. Usenet is organized into newsgroup sharing a theme or topic, with the top of the hierarchy being *comp, humanities, misc, news, rec, sci, soc, talk*. For example the usenet for the BASH shell bugs is “gnu.bash.bug”. The modern internet culture of memes started within usenet.
- **Tim Berners-Lee** The person who published the very first website on the 20th of December 1990. With a bachelors degree from *The Queens Collage, Oxford* graduating in 1976. By 1980 he was working as an independent contractor at CERN (European Organization for Nuclear Research). It wasn't until 1984 that he was a fellow at CERN. It was there that he developed the HTTPd (Hyper Text Transfer Protocol daemon), The First web browser and the first HTML (Hyper Text Markup Language) document that became the first website. In 1994 at MIT he founded the W3c (The World Wide Web Consortium) which decided the standards for the growing internet.
- **Y2K** Known as ‘*The Year 200 Problem*’ was a potential problem for computers due to the way data relating to dates is stored in computers. Many system and applications stored the year as only the last two digits thereby making 2000 and 1900 the same value, to a computer. The public and media hyped up Y2K as some potential apocalypse with fears of planes dropping out the sky and nuclear weapons firing uncontrolled. Despite these fears nothing really notable happened due to the tireless work of developers in the years and months leading up to December 2000. A similar problems are due in the years 2042, 2100 , and 10,000.
- **Web 2.0** refers to the major change to the way the internet was designed and used. Prior to this philosophical shift the internet was made up of static pages viewed passively by end users, While images existed due to the technology of the time they where very small, compressed and of little quality. Audio and Video was largely unheard of. A web2.0 experience is built of user generated content and interactive complex websites and the birth of social media. Multimedia sharing including videos and audio over competing social networks. While the term web3.0 exist to refer to the modern semantic web, The biggest contrast can be seen between any pre-2000's website and one build in the 2010's
- **Dot-com Bubble** was a Stock Market crash caused by internet related companies being over valued and excessive speculation on the tech industry. during 1995 to 2000 the Nasdaq index rose 400%. Fueled by the 1990's

economic boom due to the fall of the Soviet Union. Venture capital flooded the tech sector. Anythin with com in its name saw its stock price rise exponentially during this period. The bubble finally burst in October 2002 fell over 78% from its peak causing a recession and numerous companies to shutdown or file bankruptcy. The survivors are household names with the likes of Amazon and Google.

Q02

- *Packets*
- *IP Address (IPv4 & IPv6)*
- *Routers & Routing*
- *Domains & DNS*

Q03

- *TCP*
- *HTTP & HTTPS*
- *Web Browsers*

Q04

- **Array:** An array is an integer indexed ordered list of elements. An array is created with “*Array.new*” or “*array = [x, 1, "hello"]*” using square bracket notation. Arrays are used when a developer wishes to collect data under a single name or when the order of elements is important.
- **Hash** A hash is an unordered list using key-value pairs. Meaning that each element is accesd via its associated key instead of an integer. A hash is created with “*Hash.new*” or “*hash = {:name => "Adam", :age => "30"}*” using curly brace notation. Hashes are used to store elements with the associated name similiar to the structre of a dictionary
- **Set** A set is an unordered values without duplicates. And has many attributes with both hashes and arrays.

Q05

Q06

- **Ruby:** is an open-source, general purpose, high-level, dynamically typed, and interpreted language. Used primarily in web development with the Rails framework.
 - **Benefits**
 - Designed with human readable syntax for ease of development
 - Open-source allowing the developer community to work on features and bug fixes
 - Being Object Orientated ruby has the benefit of modularity.
 - **Drawbacks**
 - Ruby has very little support for creating Graphical User Interfaces. While there exists some gems that attempt to implement better gui support none are widely adopted.
 - Interpreted languages like ruby are slow at runtime due to the interpreter needing to facilitate the code execution.
- **Go** is an open-source, compiled, low-level, statically typed language developed by google. Used primarily in API and back end development.
 - **Benefits**
 - As a compiled language go is very fast
 - Designed with ease of use and developer efficiency
 - A minimalist language without unneeded parts
 - **Drawbacks**
 - As a relatively new language it lacks the support and ecosystems of older mature languages
 - Simplicity and minimalism is not always a good thing in something as complex as a programming language

Q07

- **Trading Crypto-currencies:** Crypto currencies are a non physical digital currency invented in 2009 with bitcoin by “*Satoshi Nakamoto*” designed to be a decentralised alternative to government backed fiat currency. With anonymizing and encryption technology built into its core. Crypto assets are posing ethical issues for developers and non developers alike. While the “*Anti-Money Laundering and Counter-Terrorism Financing Amendment Act 2017*” adds crypto to the scope of police powers. Developers must be careful when working with blockchain.
- **Tracking Data** The GDPR (General Data Protection Regulation) is a major piece of EU legislation that came into effect in 2018. Aimed at big tech and the growing concern over data privacy. Despite being only related to the EU it had major repercussions to the global tech sector. The GDPR gives strong protections to internet users and clear guidelines on how websites and applications can use cookies and user data. And is a major factor in tech giants moving away from ‘*Third Party Cookies*’
- **Case Study: Dogecoin** What started out as a joke among crypto

enthusiasts lead to a worrying example or the ethical quagmire developers have with content algorithms. A whale is someone who has a large stake in an asset, and a pump and dump is the process of hyping up an asset so the price rises then coordinating a sell off at its peak for maximum gains. An internet influencer like Elon Musk has the wealth to hoard massive quantities of Dogecoin and can influence the price with a single tweet. He posts a meme, it trends on twitter and there is a clear correlation in price fluctuations.

Q08

'Control Flow' is a term within computer science to refer to the order with which instructions within an imperative program are executed and evaluated. In the **Ruby** programming language this is achieved with; *If Statements, Case Statements, Loops (while, until for)*, and the *each method*. For example;

```
x = gets.to_i
y = gets.to_i

if x > y then
  puts "#{x} is greater"
elsif y > x then
  puts "#{y} is greater"
else
  puts "#{x} and #{y} are equal"
end
```

In this simple script after the program takes the two inputs from the user it then encounters the beginning of the control flow at the *if statement* the next line to be executed is dependant on the values of x & y. The machine has 3 branches of options to evaluate, choosing only one.

Q09

Type Coercion is the methods used to transform one data type into another. In ruby two of these methods are;

- **to_i** which returns the object as an integer
- **to_s** which returns the object as a string

These methods are used throughout ruby code from getting user input to moving around arrays and the data within them.

Q10

The ‘*Ruby Programming Language*’ like most languages has a number of different data types. Namely ‘*Numbers (Integers & Floats)*’, ‘*Strings*’, ‘*Boolean*’, ‘*Symbols*’, ‘*Arrays*’, and ‘*Hashes*’. The simplest types are numbers and strings;

```
8, 3.14159, "Hello World"
```

These are examples of integers, floating point numbers or decimal numbers and a string. The more complex types are ‘*Arrays*’ and ‘*Hashes*’;

```
array = [1, 2, 3]
hash = {:a => 1, :b => 2, :c => 3}
```

Arrays are an ordered set of data indexed at 0. To access an array the syntax is;

```
puts array[1]
```

which will print out the second element *2*. Whereas a *hash* is collection of unique keys and values. To access a hash the syntax is;

```
puts hash[:c]
```

which will print out the third element *3*. The key difference between arrays and hashes is, A hash may use any object as its key whereas an array only uses the assigned integer.

Q11

I would use the following classes;

- ***Menu***: Holds the information on each food item including name, price and availability
- ***Customer***: Able to create new or retrieve existing customer details including address and payment
- ***Order***: Stores information on which customer orders what items.
- ***Delivery***: Customer and Order information for delivery.

Q12

After writing out the ruby code from question 12 the first attempt at running it produced the following error.

```
q12.rb:2:in '<main>': undefined method '/' for...
```

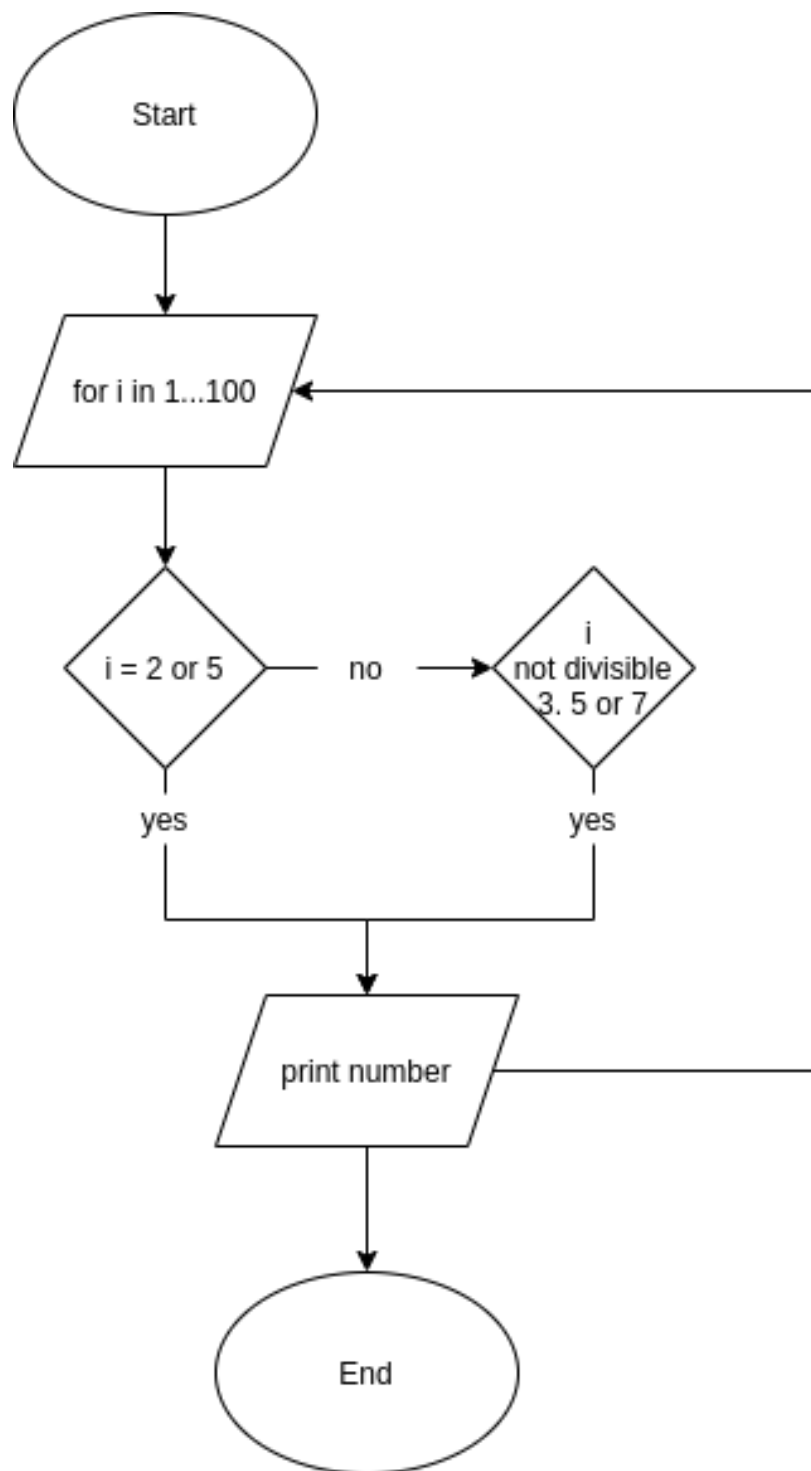
Changing line one from “celsius = gets” to “celsius = gets.to_f” produced the correct output. The gets method receives the users input saved as the variable celsius. When the user enters a number the gets method receives from standard input a string. The calculation on line 2 can not be done to strings. The method to_f returns the result of gets as a floating point number which will

allow the calculation to run and prevent any rounding errors that would have been produced by `to_i`.

Q13

```
arr = [5, 22, 29, 39, 19, 51, 78, 96, 84]
i = 0
while (i < arr.size - 1 && arr[i] < arr[i + 1])
  i = i + 1
end
x = arr[i]
arr[i] = arr[i + 1]
arr[i + 1] = x
puts arr
```

Q14



Pseudocode

```
loop start
  range 1 to 100
  if number equals 2 or 5
    print the number
  otherwise if the number is od and not 1
    if number is not divisible by 5, 3, and 7
      print the number
Loop End
```

Ruby Code

```
for i in 1..100 do
  if i == 2 || i == 5
    puts i
  elsif i.odd? && i != 1
    if i % 5 != 0 && i % 3 != 0 && i % 7 != 0
      puts i
    end
  end
end
end
```

Q15

```
puts "weather checker"
puts "Whats the Temp in c"
temp = gets.to_i
puts "Is it raning yes/no"
rain = gets.chomp
```

```
case rain
when "y", "Y", "yes", "Yes"
  rain = true
when "n", "N", "no", "No"
  rain = false
end
```

```
if rain == true && temp < 15
  puts "It's wet and cold"
elsif rain == false && temp < 15
  puts "It's not raining but cold"
elsif rain == false && temp >= 15
  puts "It's warm but not raining"
else
  puts "It's warm and raining"
end
```

Q16

```
skills = ["Python", "Ruby", "Bash", "Git", "HTML", "TDD", "CSS", "JavaScript"]
skillValue = [1,2,4,8,16,32,64,128]
userScore = Array.new
codeScore = Array.new

for i in 0...skills.length do
  puts "What is your skill in #{skills[i]}"
  ans = gets.chomp.to_i
  userScore.push(ans)
end

for i in 0...userScore.length do
  val1 = skillValue[i]
  val2 = userScore[i]
  total = val1 * val2
  codeScore.push(total)
end

totalScore = codeScore.sum

skillWeight = Hash[skills.zip(skillValue)]
result = Hash[skills.zip(codeScore)]

puts "-----"
puts "Your total score is #{totalScore}"
puts "-----"
puts "Your lowest skill is #{codeScore.min}"
puts "-----"
result.each do |l, s|
  puts "#{l} : #{s}"
end
puts "-----"
puts "The weight for each skill is"
skillWeight.each do |s, v|
  puts "#{s} : #{v}"
end
puts "-----"
```