

Q1. Identify and explain common and important components and concepts of web development markup languages

Ans: First and foremost, we must understand the word "markup language." A markup language is a human interpreted language that a software needs to mark a digital document in order for it to better grasp the content's style and structure. Markup languages are used to describe items of data about a text's content, as evidenced by the phrase "markup." In the past, the word was applied to typesetting, where instructions were "marked up" alongside the type (Ranganathan 2010).

The lengthy versions of certain markup languages are as follows:

HTML is an acronym for Hypertext Markup Language.

KML stands for "Key Whole Markup Language."

MathML (Mathematical Markup Language) is a mathematical markup language.

SGML is an acronym for Standard Generalized Markup Language. extensible Hypertext Markup Language (XHTML) extensible Markup Language (XML) is a markup language

When it comes to markup languages, there are so many alternatives that describing them all in a short article would be difficult. HTML, XML, and XHTML are the most common markup languages used in online design and development .

These markup languages are popular among web developers since they are diverse and contemporary, simple to follow, and interoperable with present tech.

HTML: The cornerstone of the World Wide Web is this language. It is related to SGML in certain ways, but this should not be confused with it. The structure of a web document is defined using HTML. HTML makes use of tags and attributes to do this. To rapidly locate the tags and properties you want, use the HTML Cheat Sheet (Web Workers).

The element usually starts the structure of an HTML document. It then goes on to include the tag. It should go without saying that you should put your page's title between this tag and the one after it, which is. When a / symbol is added to a tag, it signifies the end of the tag. For example, when appears, it will come to an end. The document comes to a finish with. It makes it possible to connect files using hypertexts and build interactive documents by utilising specific HTML tag features. HTML is also known as "website code" by others. In reality, no coding is required. Even though the words are commonly used indiscriminately, HTML is merely a markup language.

Xml- The second most prevalent markup language is XML, which stands for extensible Markup Language. In reality, XML is a type of HTML extension. The languages are complementary to one another. Although there is a significant difference between XML and HTML, consumers may quickly adapt to the new format. Rather than tags, which are used in HTML, XML employs nodes. These nodes are data items. Data may be organised using XML tags. For example, if you own a restaurant website, XML tags may be used to arrange the menu, while HTML can be used to customise how it looks. As previously stated, the two markup languages work together to achieve various goals (Web Workers).

XHTML- Ultimately, extensible Hypertext Markup Language (XHTML) is a hybrid of HTML and XML. A merger between them was a suitable decision since it was evident that both were necessary for a practical conclusion. That's how XHTML came to be.

It was developed prior to the advent of HTML5 and was initially developed for Net device screens. XHTML combines the best of both worlds to provide a robust, self-contained markup language.

Developers can add their own objects or tags to XHTML and utilise them in their sites, making it more versatile. It's simple to use and allows users more flexibility over their designs, both in terms of appearance and structure.

Some components of Markup language:

Tags- A tag is accountable for indicating the start and end of an item. They come in pairs and are made up of greater-than and less-than symbols separated by letters that define the commands. Whenever the tags are closed, there is a reverse slashing. Tags are used in practically all markup languages to provide directions on both the contents and appearance of a content (Microsoft Silverlight 2010).

Elements-Tags indicate where an element starts and finishes; a component includes both tags as well as the content placed within both.

Web forms- Web forms allow a standard browser and a web host to communicate. The data user - supplied in the forms is transferred to a server, which reacts to the values received (e.g. returns the result of a search). However, the number of widgets available for usage in forms is restricted.

Microdata- Microdata is a technology that makes information in HTML publications that is primarily intended for end users (contact information, location information, and so on) machine-readable and hence suitable for analysis and processing (Microsoft Silverlight 2010).

Cascading Style Sheet (CSS)- The markup technologies are largely used to specify the underlying basis of digital materials and sites, whereas CSS – a style sheet language that specifies appearance principles – is typically used to determine the final presentation and formatting. It is used to make the website more attractive to the clients and make the website more responsive and understandable. Perspectives and Boundaries is a program which allows you to utilise numerous themes that may be scaled and adjusted comparative or perfectly. It allows for the reuse of photos in a variety of scenarios as well as more precise filling of certain sections. Patterns, rounded edges, reflections, and even frontier pictures are all possible with this module. The border-image estate lets you to utilise an image viewer as an entity's border. The slope attribute allows the borders or backgrounds to vary from one standard to another. (Web Workers)

Q2. Define the features of the following technologies that are essential in terms of the development of the internet: packets, IP addresses (IPv4 and IPv6), routers and routing, domains and DNS. Explain how each technology has contributed to the development of the internet.

Ans:

Packets: Data is transferred through the Internet in packets, which are called messages. But, before this communication can indeed be conveyed, it must first be broken down into several smaller pieces known as packets. According to Nimkarde (2018) the data or units of information that a base station (computing device) would like to exchange to/from the target node are referred to as network packets. These packages are supplied separately. The optimum data packet is often around 1000 and 3000 strings long. The Internet Protocol defines the packetization of information (Li, 2017). Packets appear to be involved in every action you do on the Internet. So, each Web page users obtain, is made up of packets, and every e-mail you write is made up of packets as well. Packet switching network is a set-in which information is sent in tiny bits.

IP address: A computer device's Internet protocol is its location in a computer network. An IP address, in scientific jargon, is a numerical that is used to identify devices on a network. This implies that all connection in and out of that particular network would be made using its IP address. Initial computer programmers sought to assign a different identifier to each machine mostly on Internet, similar to how contact information is assigned nowadays (Nimkarde, 2018). As a result, they devised the notion of TCP/IP. In today's world, there are two types of IP addresses:

IPv4 addresses are 32 bits long (four bytes). These are easy to have and thus are being used to recognize devices all around the globe –.

IPv4 addresses are also classified into the following categories: A, B, C, D, and E (Nimkarde, 2018)..

IPv6: IPv6 addresses are very modern and consist of eight hexadecimal digits followed by the character ":". Because they are volatile, they really aren't frequently employed (Nimkarde, 2018).

Routers & Routing: A router is a physical component responsible for packet routing. A router's function is to figure out where a packet originated from and which destination node the sender node wants to transmit it to, in other words, to transport packets from source to destination. A router does this by determining the target node address of a network packet and forwarding it to that address. The "Routing Protocol" explains how routers interact with one another. Routers also create a "Routing

Table," which determines the most efficient network pathways for transmitting packets (Nimkarde, 2018). The packet is delivered to the network that has the IP address if it is detected. If this is not the case, the router passes the packet down the backbone hierarchy to the next router (Shuler, 2002).

Domains and DNS: Users require a system that transforms domain names to IP addresses and back in order to access Internet resources using user-friendly domain names rather than IP addresses. The Domain Name System (DNS) is the engine responsible for this translation (Chandramouli and Rose, 2013). A Domain Name Server, according to Nimkarde (2018), is a server with a large database of domain name mapping IP addresses that searches for the domain input and provides the IP address of the machine hosting the website you wish to visit. There are three pieces to a domain: the protocol (HTTP), the domain name, and the domain extension (generic top level such as .org, .net; and country code such as: .au).

Q3: Define the features of the following technologies that are essential in terms of the development of the internet:

TCP is in charge of routing application protocols to the proper target computer application. In the protocol stack, the TCP layer is placed beneath the application layer. Whenever applications establish a contact to another computer on the Internet, the information they provide (using a

specific application layer protocol) are sent along the network to the TCP layer. This is accomplished through the use of port numbers. The ports on each computer may be thought of as separate channels. When a packet arrives at a computer and makes its way up the protocol stack, the TCP layer chooses which application gets it depending on the port number destination (Shuler, 2002). TCP is a connection-oriented, reliable byte streaming protocol, which means that two TCP-enabled programmes must first establish a connection before sending data. TCP is dependable because it sends an acknowledging to the recipient for each packet received, confirming transmission (Shuler, 2002).

Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS) are the software protocols that enable the internet operate. HTTP is the Internet protocol used by web browsers and web servers to interact with one another. It's called an application-level protocol because it sits on top of the TCP layer in the protocol stack and is used to communicate between specific uses, such as browser and the website (Shuler, 2002).

The majority of algorithms are correlation, which means that the two computers conversing with each other maintain the Internet connection open. This is not the case with HTTP, which is a text-based protocol that does not require a connection. Requests for online components such as web pages and pictures are sent to web servers by web browsers. When a server responds to a request, the connection between the browser and the server over the Internet is broken. Each request necessitates the establishment of a new network (Shuler, 2002).

Web browsers: Internet browsers manage all inquiries, processing, and debugging tools. A web browser's core components include the interface, browsing algorithm, graphics engine, connectivity, UI backend, JavaScript parser, and storage systems (Irish and Garsiel, 2011).

The fundamental function of a browser is to display the online resource you choose by requesting it from the server and displaying it in the browser window. A browser will frequently request HTML, CSS, JavaScript, and picture content from a server and parse it according to web standards (<https://www.w3.org/standards>). These specifications allow webpages to behave reliably across browsers (Irish and Garsiel, 2011). The following UI components are found in just about all web user interface design: an address bar for entering a URL, back and forwards buttons, bookmarking choices, reload and stop buttons for renewing or pausing the loading of current documents, and a home button that leads you to your home page (Irish and Garsiel, 2011).

Developer tools: Web developers may use developer techniques to measure and troubleshoot existing script. They vary from website owners and interconnected development platforms (IDEs) in that they are tools for evaluating the user interface of a website or web application, rather than assisting in the direct building of a webpage (Irish and Garsiel, 2011).

Web development tools are available as add-ons for web browsers or as built-in functionality in web browsers. Most prominent web browsers, such as Google Chrome, Firefox, Internet Explorer, Safari, and Opera, include constructed web developer capabilities, and many more add-ons can be obtained in their individual component downloading centres (Irish and Garsiel, 2011).

Developers can use web development tools to deal with a range of web technologies, such as HTML, CSS, the DOM, and JavaScript.

Q4: Identify THREE data structures used in the Ruby programming language and explain the reasons for using each.

Data structures are strategies to organise and access data in specified ways (Castello, 2019). Add, remove, find, or sort some or all data pieces are some of the most typical actions you'll want to accomplish with data. You'll need to pick a data structure that makes the process easier, quicker, and more efficient, depending on the task at hand (Urie, 2018).

"Objects" are elements (data values) that the structure stores in object-oriented programming languages like Ruby (Urie, 2018). The following are three data structures that are often used in Ruby:

1. Arrays are a continuous collection of values of the same type that have a defined length and are placed in contiguous memory regions. Arrays hold items in a continuous slice of memory, one after the other, with no gaps between them. Elements are the values stored in an array, and they are accessible by their index, which is an integer number that indicates the element's ordinal value (Fox, 2011).
Arrays are useful because they allow for extremely quick access. This is achievable because: (a) all elements in an array are of the same kind, requiring the same amount of memory to store, and (b) the elements are stored in contiguous memory regions (Fox, 2011). In general, you should use an array to: gather things, have rapid access to items and know their index, have a pretty consistent number of items, or know a maximum amount ahead of time, need to sort and search quickly, and won't need to insert or delete data regularly (Castello, 2019; Urie, 2018).
2. Hashes (also known as associative arrays, maps, or dictionaries) are indexed collections of object references, similar to arrays. While you can only index arrays with numbers, you may index hashes with any form of object, including strings and regular expressions. In order to save a value in a hash, you must provide two objects: the index (also known as the key) and the value. The value may then be retrieved by indexing the hash with the same key. A hash's values can be any sort of object. Hashes offer one big benefit over arrays: they may use any object as an index (Talim, S, 2016).
A hash is commonly used for counting letters in a string and translating words to numbers, and to search replications in an array.
3. Trees, unlike the previous two data structures, grow and decrease in length and depth. Trees in Ruby, as in real life, start with a root and then branch out. However, the tree data structure is frequently portrayed as starting at the top and working its way down. Hubs in trees can contain both values and pointers to their offspring nodes. Tree nodes can contain any number of children nodes, although two is the most typical. Binary trees are trees with a maximum of two nodes in each node (Urie, 2018). Trees are typically employed for data that has to be categorized and has a classification. When binary trees are maintained ordered, the value of the left leaf nodes will be less than the value of the parent node, and the value of the right child node is higher. This makes it incredibly quick and straightforward to identify a certain value without having to compare it to all of the other possibilities (Urie, 2018). Trees are also useful for storing large volumes of data that must be swiftly sorted and found. Trees are a wonderful alternative if the data will change often yet still be easily searchable (Urie, 2018).

Q5: Describe the features of interpreters and compilers and how they are different.

A compiler is a piece of software that converts code written in a high-level programming language to machine code. It's essentially a software that converts human-readable code into a language that a computer processor can comprehend (binary 1 and 0 bits). The computer then executes the bytecode in order to complete the tasks.

Likewise, an interpreter is a computer software that converts high-level programme statements, such as source code, pre-compiled code, and scripting, into assembly language. As you can see, both the compiler and the interpreter are responsible for translating higher-level programming languages into machine code. The interpreter converts code into machine code when the programme runs, whereas the compiler converts code into machine code before the programme runs.

On the basis of their major characteristics, there are also some distinctions between compilers and interpreters:

Steps in Programming: A compiler must first create the programme, then parse or analyse all of the language statements for accuracy. Convert source code to machine code if there are no errors. The compiler then assembles the various code files into an executable programme (exe) and finally executes the programme. Interpreters, on the other hand, have fewer programme phases, which include: generating the programme. There is no file linking or machine code creation. Finally, throughout the execution, source statements were performed line by line.

Machine code: Compilers write assembly language to disc as bytecode. On the other hand, interpreters need not preserve any assembly language.

Compilation speed: compiled code runs quicker. The speed of interpreted code is slower. The language transcription paradigm is used by compilers. The Interpretation Method is used to choose interpreters.

Compilers provide output programmes (in the form of exe files) that may be executed independently of the source programme. Interpreters do not produce output programmes; instead, they assess the source programme at each step of the execution process.

Program execution: After the full output programme has been compiled, the programme is executed. However, because programming is a component of the analysis phase, it is done point by point.

Storage prerequisites: The target programme runs on its own and does not require the compiler to be loaded into memory. During interpretation, the interpreter resides in the memory.

The compiler reads the complete code before optimising it. As a result, it does several optimizations to make code run quicker. Because interpreters read code line by line, optimisation are less reliable than with compilers.

Speed: The compilation process is rather complex, and evaluating and processing the programme takes a substantial amount of time. On the other hand, interpreters devote less time to studying and processing the programme.

Mistakes: show all faults at the very same period after synthesis. Each line's fault is displayed by the interpreter one by one (guru.99, n.d).

Q6: Identify TWO commonly used programming languages and explain the benefits and drawbacks of each.

A programming language is a formal language that describes a collection of instructions or directions that may be used to produce various types of computer outputs. A machine language, often known

as object code, is the most basic sort of computer language. A collection of binary code that is specific to the kind of CPU is known as object code. Each object code instruction corresponds to a CPU basic operation, necessitating a large number of code instructions. High-level programming languages, on the other hand, are programming languages that individuals can write in a natural way (Humer and Foster, 2014).

There are hundreds or thousands of elevated programming languages available. Java and Python are ranked first and second, respectively, in the 2019 TIOBE index, one of the top listings of widely used programming languages (<https://www.tiobe.com/tiobe-index/>).

JAVA- Sun Microsystems, which was founded by James Gosling in June 1991, was the first to design the Java programming language. It was first introduced in 1994 under the name Oak, with the intention of being utilised in integrated consumer electrical gadgets. It promised to be WORA (Write Once, Run Anywhere), with free run-times on prominent platforms. The technology was nicknamed Java and updated for producing Websites and applications a year later (Humer and Foster, 2014). Java is a programming language that focuses on objects. Everything in Java is considered an object since it is an extension of the object model. Java is also platform agnostic. When Java is compiled, unlike other programming languages, it is converted into platform-independent byte code rather than platform-specific machine code. This byte code may be delivered over the internet to any machine, which is then interpreted by the Java Virtual Machine (JVM) on the platform being used. Java is supposed to be simple to learn, which is why it is regarded as the ideal language for programmers and developers to learn.

Java is also safe, as it allows for the creation of virus-free, tamper-proof systems and uses public-key encryption for authentication. With the inclusion of a Java runtime system, the Java compiler creates an architecture-neutral object file format, allowing the produced code to operate on a variety of processors. Java is portable because it is architecturally agnostic and lacks implementation-specific features. Java makes an attempt to reduce error-prone situations by focusing on compile-time and run-time error checking, making it a robust programming language (Humer and Foster, 2014). Because Java is multithreaded, it is feasible to design programmes that can do many tasks at the same time. This design element enables developers to create interactive apps that operate seamlessly. The development process is also more speedy and analytical since Java byte code is transformed to native machine instructions and is not stored anywhere (Humer and Foster, 2014).

Java has the following advantages: It was meant to be easier to use, develop, build, and debug than other languages. Creates reusable code and standard programmes. Has the capacity to transfer easily from one automated data source to another. Allows one processing system to communicate with another. Some of its drawbacks include much higher memory usage than C or C++. The appearance and feel of GUI programmes by default. Languages with a single paradigm (Naveen Reddy, Geyavalli, Sujani, and Rajesh, 2018).

Python Guido van Rossum created the Python programming language at the Netherlands' National Research Institute for Mathematics and Computer Science in the late 1980s and early 1990s. Python is based on various scripting languages such as ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and others (Humer and Foster, 2014).

PYTHON- is one of the most widely used dynamic programming languages today. It is a generalpurpose programming language that is sometimes mistaken for a scripting language. It's a design concept that prioritises code readability and makes liberal use of white space. Python creates logical programming on a local and large scale. Python is an object-oriented programming language

with a run time of 71.90 seconds and a memory footprint of 71.90 bytes and the storage utilisation is about 2.80mb/sec (Naveen Reddy, Geyavalli, Sujani, and Rajesh, 2018).

Python's key characteristics include the following. Python is simple to learn because it has a small number of keywords, a straightforward structure, and a well-defined syntax. Python is also noted for its readability. Python code is more clearer and more visible to the naked eye. Python's popularity may possibly be due to the fact that its source code is very simple to maintain. The majority of Python's library is relatively portable and cross-platform compatible on Unix, Windows, and Macintosh, which is one of the language's biggest assets. Python also has an interactive mode, which allows you to enter results from a terminal directly into the language, allowing you to test and debug code snippets in real time. Python is portable, meaning it can operate on a broad range of hardware systems while maintaining the same user interface. Python is expandable, allowing you to add lowlevel modules to the Python interpreter, increasing the efficiency of developer tools. Python facilitates the creation and porting of graphical user interfaces to a variety of system calls, libraries, and operating systems, including Windows MFC, Macintosh, and Unix's X Window system. Last but not least, Python is more scalable than shell programming and provides a better structure and support for huge projects (Humer and Foster, 2014). Python has several advantages, such as extensive libraries. Productivity has increased. It's open source and free. Speed limits, an immature data base access layer, and design restrictions are some of the downsides (Naveen Reddy, Geyavalli, Sujani, and Rajesh) (2018).

Q7: Identify TWO ethical issues from the areas below and discuss the extent to which an IT professional is ethically responsible in terms of the issue.

List of topics containing ethical issues:

- access to a user's personal information (medical, family, financial, personal attributes such as ## sexuality, religion, or beliefs)
- intellectual property, copyright, and acknowledgement.
- criminal acts such as theft, fraud, trafficking and distribution of prohibited substances, terrorism
- GPS tracking data and other types of metadata, MAC addresses, hardware fingerprints
- freedom of thought, conscience, speech and the media
- aggressive sales and marketing practices designed to mislead and deceive consumers
- trading of shares on the stock exchange OR crypto-currencies

For each ethical issue identify a source of legal information relating to the ethical issue and discuss whether the law is helpful in assisting a developer to act in an ethical way.

Conduct research into a case study of ONE of the ethical issues you have chosen discuss how an ethical IT professional should respond to the case study and how they might mitigate or prevent ethical breaches.

An ethical decision is one that is founded on contemplation on what matters to a person and is consistent with those convictions. The law, on the other hand, aims to establish a fundamental, enforceable standard of behaviour that is required for a society to prosper and in which everyone is

treated fairly (The Ethics Centre, 2016). Let's look at two of the scenarios from the list above and see what ethical and legal difficulties they raise.

Access to the private information of a user:

The ethical considerations that apply to this scenario have to do with privacy and confidentiality, which are two words that are related but not identical. The term "privacy" refers to a personal state of being shielded from prying eyes (Neetling et al., 1996, p. 36). When we say that information is confidential, we mean that we expect it to be shared only with those who have been given permission, and only with those who have been given permission. Privacy is a crucial human right since it is a prerequisite for other rights like freedom and personal autonomy (Britz, n. d). Developers have access to and alter vast volumes of data and information due to the nature of their employment. This openness entails a great deal of responsibility. Developers must respect and defend the privacy and confidentiality of their users and clients, according to the Australian Computer Society's Code of Ethics.

When developers adhere to the idea of public interest primacy in their work, they recognise that the public interest trumps personal, private, and sectional interests, and that any conflicts should be addressed in favour of the public interest. They accomplish this by: identifying those who may be impacted by your work and explicitly considering their preferences; maintaining the security and anonymity of others' data; raising any potential issues among your chosen profession and lawful or other acknowledged example by having with interested parties; and network partners as quickly as possible of any improprieties or observant challenges that you have. When developers follow this approach, they are less likely to violate their customers' and users' privacy and confidentiality. Furthermore, the language encourages developers to follow the idea of truthfulness. In this regard, integrity is not betraying the public's faith in the industry or the confidence of your various stakeholder groups. Engineers as well as other computer scientists really shouldn't wilfully deceive a customer or prospective customer about the acceptability of a product or service, according to this concept (ACS, 2014a). If programmers aim for integrity in their job, they are less likely to violate their customers' and customers' relating to confidentiality.

Confidentiality is seen as a fundamental right in Australia, and the Privacy Act protects it (1988). The Privacy Act is founded on 13 Australian privacy principles that apply to most Australian government bodies as well as certain private sector entities. The Australian Privacy Principles are technology agnostic, allowing them to evolve with the times.

Ownership, proprietary information, and recognition

The Internet has become one of the major breakthroughs of the contemporary world and enables rapid access to all types of essential information at one's fingertips. However, every technological advancement has both beneficial and negative consequences. Replication, unlawful streaming, accessing, duplicating, theft, and exploitation of intellectual property are some of the issues that the Internet and its accessibility face. Designers must preserve the intellectual property of everyone else, as per the ACS code of ethics. It is in the citizenry's best advantage to do just that, as stated in the preceding concept of supremacy of public interest. Without a certainty, the concept of integrity applicable to this situation. The code of ethics explicitly specifies that programmers and software engineers should give credit where it was due to others' efforts, and that they should not try to boost their personal profile at the cost of someone else's (ACS, 2014a). This case also pertains to the competency assumption. Competency implies that an individual is aware of these issues and will only take tasks that they feel they can complete. It is impossible for a single person to be entirely aware and proficient in every aspect of the industry. As a result, when components of a project

exceed your talents, the programmers should not avoid seeking extra knowledge and advice from relevant experts and hold them responsible for their efforts. Engineers and computing systems should not exaggerate their abilities or understanding, and therefore should appreciate and consult the legal experience of peers in their specialities as appropriate, according to this idea (ACS, 2014a).

In terms of legislation, Australia has the Copyright Act 1968. The originator of an intellectual, theatrical, orchestral, or artistic work holds the trademark underneath this legislation. Underneath the Copyright Act, a computer model or content inserted on the internet is considered a "literary work," where in case the developer is the holder of copyright in the computer programme (La Trobe University, 2014)

Case Study- Chaitee is a young programmer who is attempting to construct a massive quantitative application for her firm. This firm encourages developers to talk discuss their research and submit their strategies in scholarly publications. Chaitee has been trapped on various areas of the software despite days of arduous development. Her boss, who doesn't understand the original problem intricacy, expects the project done during the next few days.

Chaitee recalls that a colleague did provide her code files from his ongoing project and an early designs of a commercialized software platform built at some other firm when she was stumped. Chaitee notices two portions of code in these applications that may be immediately included in her own design. Chaitee uses blocks of code from her colleague as well as from the company software, however did not mean to mention anyone during the documentation. Chaitee finishes the program and submits it with a day in hand (ACS, 2014b).

The case study highlights an ethical dilemma around copyrighted material. As previously stated, the ethical norms violated here concern the primacy of public interest and competence. Sam exaggerated her abilities and expertise by utilising pieces of her colleague and another firm's program and failing to acknowledge it in the paperwork. While neglecting to: safeguard the copyrights of the others; appreciate and safeguard the unique shareholders' interests; and obtain technical competence from peers in their area of knowledge (ACS, 2014b).

Q8: Explain control flow, using an example from the Ruby programming language

A data flow element is a makes it understandable that interrupts the regular development to another expression by branching to another position in program code, either temporarily or irrevocably (Learn n.d.). Conditionals are by far the most frequent correspond, with the following fundamental forms: If, else, and elsif expressions, condition declarations, and looping being the most popular. Let's take a closer look into while loops. At the while command, the control flow enters the while loop. This statement decides whether or not the action will reach the loop's content. The instructions in the loop are performed if the precondition value is true. The system call will proceed in this scenario through all expressions supplied. When a statement completes its execution, the system call returns to the initial instruction (while), and the cycle will begin all over again. If the condition becomes false, the control flow proceeds to the instruction, which ends the loop (Freider, Freider, and Grossman, 2013).

Following Code:

```
n = 5 i = 0
while (i <=
0)
```

```
    puts i
    i = i + 1
end
```

This is indeed a simple programme that displays all numbers from 0 to 5. The loop iterates, printing the quantity in *i* and then increasing it by one. The loop runs as far as *i* is less than or equivalent to *n* = 5 and the result of *i* is much less than or equal to *n* = 5. The integer *i* will be increased in this scenario, and then after 6 repetitions, *i* would be greater than *n*, ending the loop (Freider, Freider, Grossman, 2013).

Q9: Explain type coercion

In computer engineering, there are various methods for transforming an object with one data type into some other, either expressly or impliedly. Type coercion is an example of one of them.

Inside an equation, type coercion is described as the automated or implied transformation of a value through one data type to another. Since the item is saved as one type of data, however the context demands a various data type, compulsion happens. Type converter and type coercion are similar in that they always change items from one data type to another, but type coercion is implicit, whereas type conversion can be either implicit or explicit (mdn web docs 2021).

Type coercion, according to McFarlin (2014), is why an interpreted or processor determines what sort of contrast is now being done, while transformation is an explicitly written shift in type by the developer.

One kind is constantly compelled to become the other, and the process usually follows a hierarchy. Although every dialect is distinct, notice that when we concatenate a text and a numeric in the second example, the output is also a string. Because the number has been converted to a string, this is the case (McFarlin, 2014).

Q10: Describe the data types recognised by the Ruby programming language. In your description you should give example code which uses each data type, and include the name of the Ruby classes which represent each data type.

To grasp a computer language, Flanagan and Matsumoto (2008) claim that you must first comprehend what types of input it can handle and what it could do with such an input. Types of data instruct the machine about how to process the information in your application. They also define all you could do with the information, such as which procedures you may conduct.

As per Hogan (2017), the forms of data that may be modified with the Ruby programming language include:

Symbol: In a Ruby application, symbols are a specific information structure that works as a tag or identify. Immutability refers to the fact that symbols cannot be modified. A symbol seems to be a constant entry, but it has no value.

Array: Arrays and hashes were discussed in further depth in question Q4. As previously stated, arrays and hashes allow people to interact with sets of information and are, in some ways, data carriers. As a result, they are frequently referred to as data architectures instead of different data, though others believe them to be both (Flanagan and Matsumoto, 2008).

Range: Ranges are the quantities that fall among a beginning and an ending measure. The origin and finish values are separated by two or three points in ranged regular expressions. The range is

inclusive if two dots are used, and the final value is included in the range. The ranging is exclusive if three dots are used, and the final value is not included in the range (Flanagan and Matsumoto, 2008).

Integers: Whole digits which can be plus, minus, or 0 (... , -1, 0, 1,...) are known as integers. Integers are also referred to as ints.

Float: A real number is represented as a floating-point number, or float. Real numbers, such as 9.0 or -116.42, could be either rational or irrational, indicating they have a decimal portion. In many other terms, in a Ruby application, a float is a number with a decimal point.

Strings: A string is a collection of characters, such as letters, numbers, and symbols, in a certain order. Single quotes (') as well as double quotes (") are the most popular spots for strings to appear ("). To make a string in Ruby, the series of characters must be surrounded in quotes.

Code explanation:

```
# explicitly assigning vales to variables
a = 3      b = 3.0      c = "Hello World"
d = :bye   e = 0..10    f = { }      g
= [ ]
```

```
# printing class of the variables
puts "#{a} : #{a.class}"      puts
"#{b} : #{b.class}"           puts
"#{c} : #{c.class}"           puts
"#{d} : #{d.class}"           puts
"#{e} : #{e.class}"           puts
"#{f} : #{f.class}"           puts
"#{g} : #{g.class}"
```

Q11: Here's the problem: "There is a restaurant serving a variety of food. The customers want to be able to buy food of their choice. All the staff just quit; how can you build an app to replace them?">

Identify the classes you would use to solve the problem

Write a short explanation of why you would use the classes you have identified.

The MVP methodology for application development to create this imaginary restaurant app. Model, View, and Controller (MVP) is an acronym for Model, View, and Controller. MVC is a common approach of coding organisation. The core concept underlying MVC is that each portion of our software has an unique function. A few of our code keeps your app's data, a few of your program makes it seem great, but some of your code regulates how it works.

The Model provides the important elements of our project, by this structure. For example, in this scenario, the Model would most likely contain a component which might create the meal and monitor the dish's condition. All of the methods that actively interact with users are found in the View subsection. This is the logic that specifies what our consumer expects and engages with our application, and also how it looks. It would most likely involve befriending farewell to the client, presenting food selections and choices, as well as other UI/UX aspects in this case. Finally, the Controller software serves as a link between the Model and the View, accepting input from the user and selecting how to respond. It's the user's brain, and it connects the model and the view (Code

Academy, 2019). In this case, I'd include the functions and methods for receiving orders, adding additional commands, splitting fees, and so on.

Class Chef (Model): Chef 1,2... n will be objects in this class. Variables like name of chef, order number, order time, order status. The class's actions or procedures would've been: these functions would seek to alter the object's states according to the characteristics specified above. In this situation, there may be ways to modify the chef's name, the order, the completion time, and status based on the order/food item ordered.

Customer (View) class: Customer 1,2,3... are the objects or instances of this class. Customer names, view menu, order placed by customer, number of orders, and so on are the properties or local variables of this class. This class's functions or methods include methods for changing the customer's name, altering orders by adding and deleting items, changing the number of orders, and so on.

Waiter (Controller) class: Waiter 1,2,3... are the objects or instances of this class. The following characteristics or instance variables would be found within this class: waiter's name, start shift time, finish shift time, show orders taken, number of orders taken, and so on. This class types methods or procedures include those that help to change the waiter's identity, respective beginning and finish shift times, the orders they've handled that day, and so on.

Q12: Identify and explain the error in the following code that is preventing correct execution of the program

```
celsius = gets  fahrenheit = (celsius * 9 / 5) + 32  print "The result is: "  print fahrenheit  puts "."
```

```
1 #12 ans
2
3 puts "The temperature in celsius is:"
4 celsius = gets.chomp.to_i
5 fahrenheit = (celsius * 9 / 5) + 32
6 print "The result in fahrenheit is: #{fahrenheit}"
7 puts "."
8
```

```
chaitee27@chaitee:~/ruby$ ruby 12WB.rb
The temperature in celsius is:
15
The result in fahrenheit is: 59.
chaitee27@chaitee:~/ruby$
```

Explanation:

A no method error was reported by the program. A number of factors might account for this. To begin with, the variable Celsius is configured to be allocated a value based on the user's input, but the user also isn't solicited for input in the code. There's also a problem with the request for user input, because "gets" itself always will provide a string irrespective of if the user enters a digit or an alphabet letter, therefore "30" rather

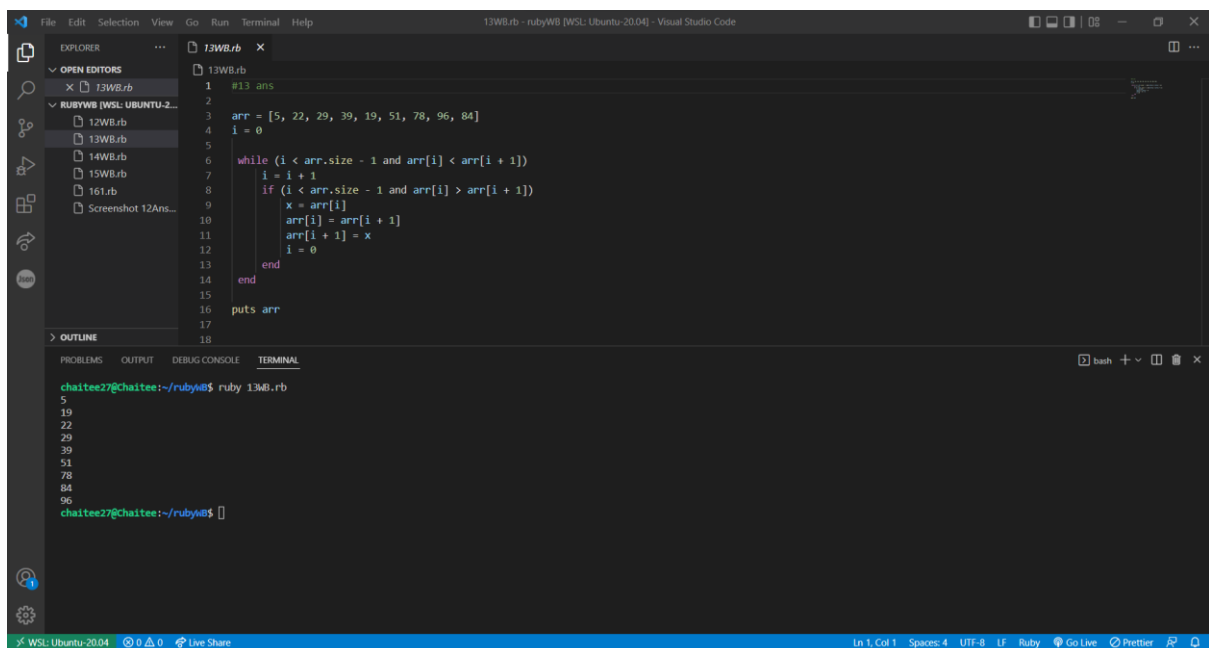
than the requested 30 will be returned. We may correct this by adding 'a.chomp.to_i' after the gets, as seen in the modified function below. This would make it possible to transform the input to an integer.

Second, while the second sentence looks to be right very first sight, it would not have been correctly performed since our initial executable contains mistakes. If Ruby believes that the value of Celsius is a string "30," it will try to increase "30" by 9, returning "303030303030303030" The software would then fail since Ruby will be unable to execute a functional form with a string, such as division.

Because the function in line two was not functional, the remaining two lines of code would be unable to output anything. I feel that correcting the initial error is adequate, but I would additionally utilise string insertion because it is a strong means of combining parameters into strings.

Q13: The following code looks for the first two elements that are out of order and swaps them; however, it is not producing the correct results. Rewrite the code so that it works correctly.

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



The screenshot shows a Visual Studio Code window with a file named '13WB.rb'. The code in the editor is as follows:

```
1 #13 ans
2
3 arr = [5, 22, 29, 39, 19, 51, 78, 96, 84]
4 i = 0
5
6 while (i < arr.size - 1 and arr[i] < arr[i + 1])
7   i = i + 1
8   if (i < arr.size - 1 and arr[i] > arr[i + 1])
9     x = arr[i]
10    arr[i] = arr[i + 1]
11    arr[i + 1] = x
12    i = 0
13  end
14 end
15
16 puts arr
17
18
```

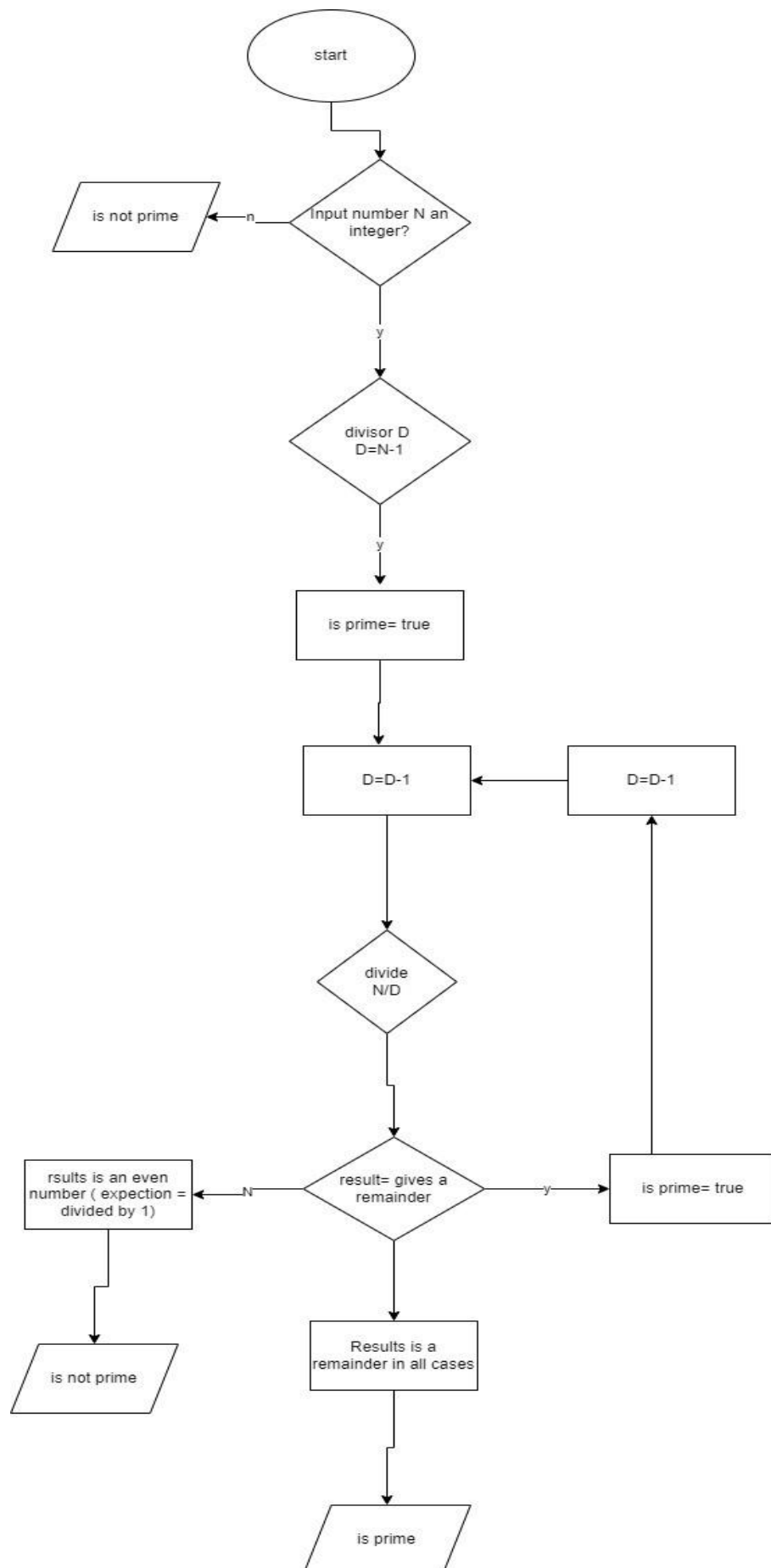
The terminal at the bottom shows the command 'ruby 13WB.rb' being executed, which outputs the array elements in order: 5, 19, 22, 29, 39, 51, 78, 84, 96.

Q14: Demonstrate your algorithmic thinking through completing the following two tasks, in order:

1. Create a flowchart to outline the steps for listing all prime numbers between 1 and 100 (inclusive). Your flowchart should make use of standard conventions for flowcharts to indicate processes, tasks, actions, or operations

2. Write pseudocode for the process outlined in your flowchart

1. flow chart:



2. Pseudocode

#A is given a number between 1 to 100. Checking an integer or not

#B is the divisor, and it is seassigned to A-1.

(#) At the start of the programme we assume is prime.

(#) To divide A % B, we establish a while loop with the condition that while B!= 1.

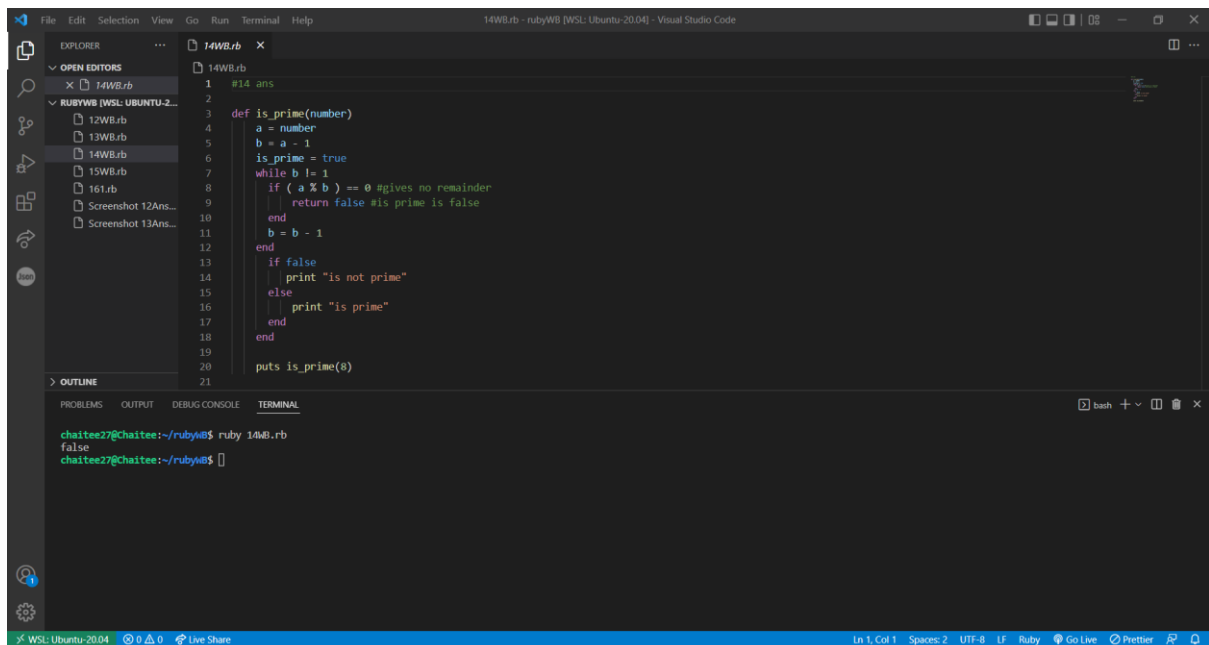
(#) It is true that is prime if the result of dividing A percent B yields a residue that is not equal to 0.

(#) The loop continues while A %B gives a remainder, and B becomes B-1 each time before the loop starts again.

(#) If A % B divides equally by at least one number other than 1 and itself, true becomes false, and the if condition moves to the second. The loop finishes when it outputs "is not prime."

(#) If A % B completes the loop and never divides evenly by any integer other than itself and 1, the condition is prime is always true, and the programme outputs "is prime" and finishes.

Code

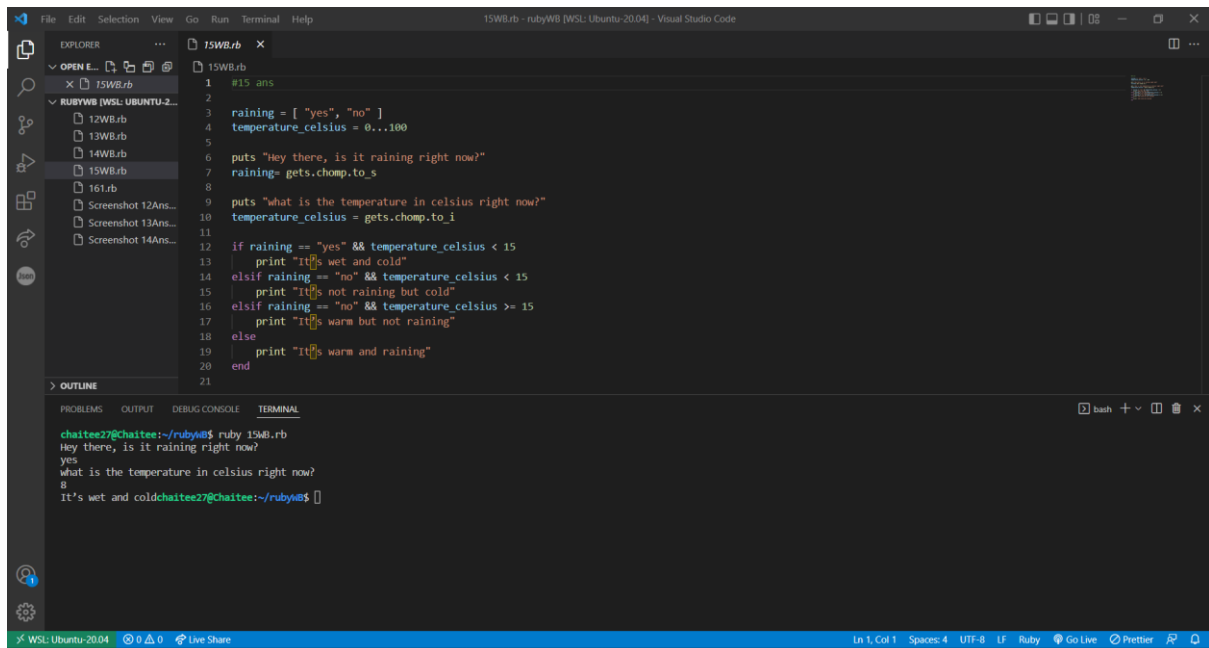


```
1 #14 ans
2
3 def is_prime(number)
4   a = number
5   b = a - 1
6   is_prime = true
7   while b != 1
8     if (a % b) == 0 #gives no remainder
9       return false #is prime is false
10    end
11    b = b - 1
12  end
13  if false
14    print "is not prime"
15  else
16    print "is prime"
17  end
18 end
19
20 puts is_prime(8)
21
```

```
chaitee27@chaitee:~/ruby40$ ruby 14WB.rb
false
chaitee27@chaitee:~/ruby40$
```


Q15: Write pseudocode OR Ruby code for the following problem:

You have access to two variables: raining (boolean) and temperature (integer). If it's raining and the temperature is less than 15 degrees, print to the screen "It's wet and cold", if it is less than 15 but not raining print "It's not raining but cold". If it's greater than or equal to 15 but not raining print "It's warm but not raining", and otherwise tell them "It's warm and raining".



The screenshot shows a Visual Studio Code editor window with a file named `15WB.rb`. The code is a Ruby script that takes two inputs from the user: a string for whether it is raining and an integer for the temperature in Celsius. It then uses conditional logic to print a message based on these inputs. The terminal at the bottom shows the execution of the script, with the user's inputs and the program's output.

```
1 #15 ans
2
3 raining = [ "yes", "no" ]
4 temperature_celsius = 0...100
5
6 puts "Hey there, is it raining right now?"
7 raining= gets.chomp.to_s
8
9 puts "what is the temperature in celsius right now?"
10 temperature_celsius = gets.chomp.to_i
11
12 if raining == "yes" && temperature_celsius < 15
13   print "It's wet and cold"
14 elsif raining == "no" && temperature_celsius < 15
15   print "It's not raining but cold"
16 elsif raining == "no" && temperature_celsius >= 15
17   print "It's warm but not raining"
18 else
19   print "It's warm and raining"
20 end
21
```

Terminal output:

```
chaitee27@chaitee:~/ruby48$ ruby 15WB.rb
Hey there, is it raining right now?
yes
what is the temperature in celsius right now?
8
It's wet and coldchaitee27@chaitee:~/ruby48$
```

Q16: ACME Corporation are hiring a new junior developer, as part of their hiring criteria they've created a "coding skill score" based on the specific competencies they require for this role; the more important the skill is for ACME corp, the more points it contributes to the "coding skill score" The skills are weighted as follows:

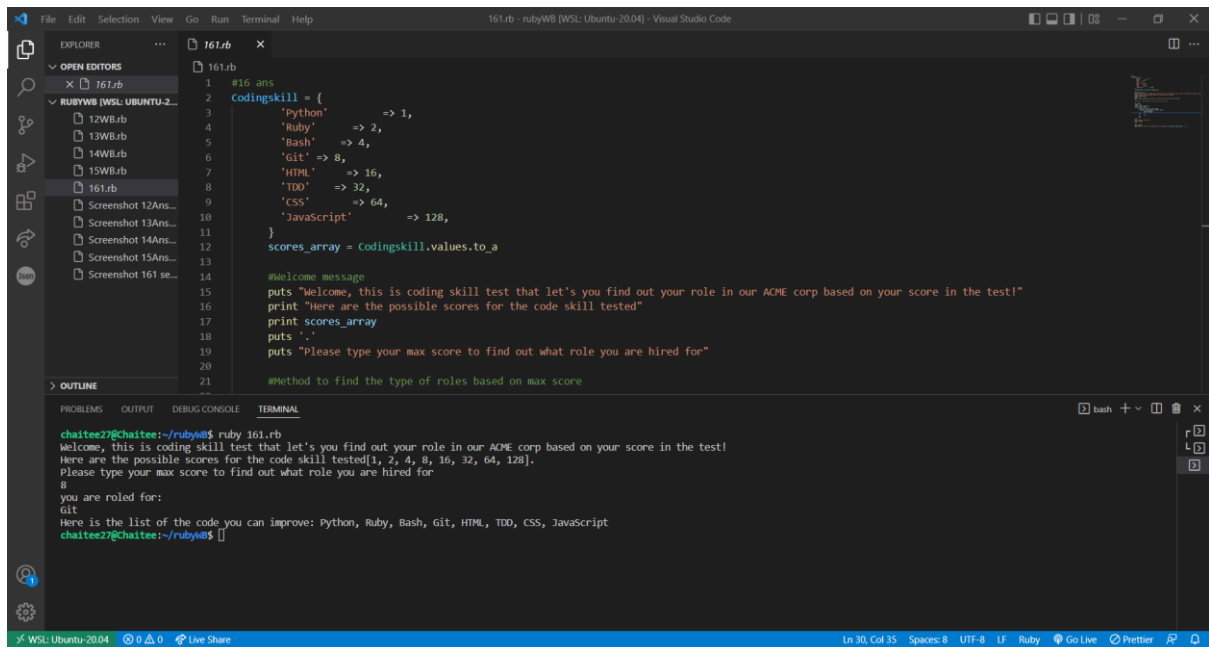
- Python (1)
- Ruby (2)
- Bash(4)
- Git (8)
- HTML (16)
- TDD (32)
- css (64)
- JavaScript (128)

Write a program that allows a user to input their skills and then tells them

- a) Their overall "coding skill score"

b) Skills they may want to learn, and how much each one would improve their score

Part 1-



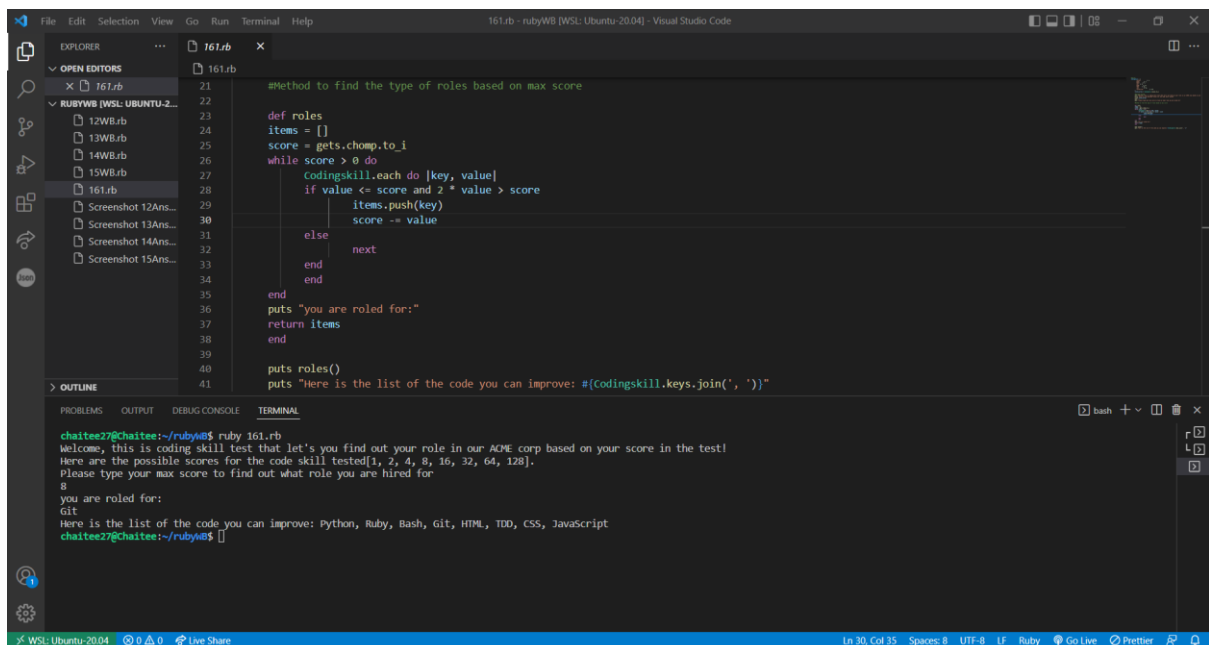
The screenshot shows the Visual Studio Code editor with a file named `161.rb` open. The file contains a Ruby script that defines a hash `codingskill` with various programming languages and their corresponding scores. It also includes a `puts` statement to display a welcome message and a list of possible scores, and a `puts` statement to prompt the user for their max score.

```
1 #16 ans
2 codingskill = {
3   'Python' => 1,
4   'Ruby'   => 2,
5   'Bash'   => 4,
6   'Git'    => 8,
7   'HTML'   => 16,
8   'TDD'    => 32,
9   'CSS'    => 64,
10  'JavaScript' => 128,
11 }
12 scores_array = Codingskill.values.to_a
13
14 #welcome message
15 puts "Welcome, this is coding skill test that let's you find out your role in our ACME corp based on your score in the test!"
16 print "Here are the possible scores for the code skill tested"
17 print scores_array
18 puts '.'
19 puts "Please type your max score to find out what role you are hired for"
20
21 #Method to find the type of roles based on max score
```

The terminal output shows the execution of the script, displaying the welcome message, the list of possible scores, and the prompt for the user's max score.

```
chaitee27@chaitee:~/rubyw8$ ruby 161.rb
Welcome, this is coding skill test that let's you find out your role in our ACME corp based on your score in the test!
Here are the possible scores for the code skill tested[1, 2, 4, 8, 16, 32, 64, 128].
Please type your max score to find out what role you are hired for
8
you are roled for:
Git
Here is the list of the code you can improve: Python, Ruby, Bash, Git, HTML, TDD, CSS, JavaScript
chaitee27@chaitee:~/rubyw8$
```

Part 2-



The screenshot shows the Visual Studio Code editor with the file `161.rb` updated. The script now includes a `def roles` method that takes a max score as input and returns a list of roles that can be improved based on that score. The terminal output shows the execution of the script, displaying the welcome message, the list of possible scores, and the prompt for the user's max score, followed by the output of the `roles` method.

```
21 #Method to find the type of roles based on max score
22
23 def roles
24   items = []
25   score = gets.chomp.to_i
26   while score > 0 do
27     codingskill.each do |key, value|
28       if value <= score and 2 * value > score
29         items.push(key)
30         score -= value
31       else
32         next
33       end
34     end
35   end
36   puts "you are roled for:"
37   return items
38 end
39
40 puts roles()
41 puts "Here is the list of the code you can improve: #{Codingskill.keys.join(', ')}"
```

The terminal output shows the execution of the script, displaying the welcome message, the list of possible scores, and the prompt for the user's max score, followed by the output of the `roles` method.

```
chaitee27@chaitee:~/rubyw8$ ruby 161.rb
Welcome, this is coding skill test that let's you find out your role in our ACME corp based on your score in the test!
Here are the possible scores for the code skill tested[1, 2, 4, 8, 16, 32, 64, 128].
Please type your max score to find out what role you are hired for
8
you are roled for:
Git
Here is the list of the code you can improve: Python, Ruby, Bash, Git, HTML, TDD, CSS, JavaScript
chaitee27@chaitee:~/rubyw8$
```

References

1. ACS (2014a). ACS Code of Professional Conduct Professional Standards Board Australian Computer Society. Retrieved from https://www.acs.org.au/content/dam/acs/rules-and-regulations/Code-of-Professional-Conduct_v2.1.pdf
2. ACS (2014b). ACS Code of Professional Conduct Case Studies. Retrieved from <https://www.acs.org.au/content/dam/acs/elected-members/pab/EthicsCommittee/ACS%20Code%20of%20Professional%20Conduct%20Case%20Studies.pdf>
3. Britz, J. J. (n.d). Technology as a threat to privacy: Ethical Challenges To The Information Profession. Retrieved From <http://web.simmons.edu/~chen/nit/NIT%2796/96-025-Britz.html>
4. Barry M. L., Cerf, V.G, Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., Postel, J., Roberts, L. G., Wolff, S. (1997). Internet Society — A Brief History of the Internet. Retrieved from https://www.internetsociety.org/wp-content/uploads/2017/09/ISOC-History-of-the-Internet_1997.pdf
5. Castello, J. (2019). An Overview of Data Structures For Ruby Developers. Retrieved from <https://www.rubyguides.com/2019/04/ruby-data-structures/>
6. Chandramouli, R. and Rose, S. (2013). Secure Domain Name System (DNS) Deployment Guide. <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81-2.pdf> Computer History Museum (2019). Internet History. Retrieved from <https://www.computerhistory.org/internethistory/1990s/>
7. Code Academy (2019). MVC: Model, View, Controller. App organization explained. Retrieved from <https://www.codecademy.com/articles/mvc>
8. Flanagan, D., & Matsumoto, Y. (2008). The Ruby Programming Language. O'Reilly Media.

9. Freider, O., Freider, G., Grossman, D., (2013). Computer Science Programming Basics in Ruby. Exploring Concepts and Curriculum with Ruby. Retrieved from <https://books.google.com.au/books?id=ff8MXr7fZq4C&pg=PA56&lpg=PA56&dq=conditional+flow+ruby&source=bl&ots=gTcXQi1TuB&sig=ACfU3U2waxZobMdOcrfaAeBHyQxgonsCpA&hl=en&sa=X&ved=2ahUKEwie9eyp6vmAhU94XMBHWixCn4Q6AEwGXoECAoQAQ#v=onepage&q=conditional%20flow%20ruby&f=false>
10. Fox, C. (2011). Concise Notes on Data Structures and Algorithms. Retrieved from https://w3.cs.jmu.edu/spragunr/CS240_F12/ConciseNotes.pdf
11. Hogan, B. (2017). Understanding Data Types in Ruby. Retrieved from <https://www.digitalocean.com/community/tutorials/understanding-datatypes-in-ruby>
<http://faculty.salina.kstate.edu/tmertz/Java/041datatypesandoperators/07typecoercionandconversion.pdf>
12. Irish, P. and Garsiel, T. (2011). How Browsers Work: Behind the scenes of modern web browsers. Retrieved from <https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/#The browsers we will talk about>
13. La Trobe University (2019). Copyright and Information Technology. Retrieved from <https://www.latrobe.edu.au/legalservices/copyright-it>
14. Li, S. (2017). How Does The Internet Work?. Retrieved from <https://medium.com/@User3141592/how-does-the-internet-workedc2e22e7eb8>
15. McFarlin, T. (2014). The Beginner's Guide to Type Coercion: What is Coercion? Retrieved from <https://code.tutsplus.com/articles/the-beginners-guide-to-type-coercion-what-is-coercion--cms-21917>.
16. Naveen Reddy, K.P, Geyavalli, Y., Sujani D., and Rajesh, S. M^[SEP] (2018). Comparison of Programming Languages: Review. International Journal of Computer Science & Communication (ISSN: 0973-7391) Volume 9 • Issue 2 pp. 113-122 March 2018 - Sept 2018. https://www.researchgate.net/publication/326672199_Comparison_of_Programming_Languages_Review
17. Nimkarde, S (2018). What computer networks are and how to actually understand them. <https://www.freecodecamp.org/news/computer-networks-and-how-to-actually-understand-them-c1401908172d/>
18. Stephen J. Humer, S.J. & Foster, E.C. (2014). A Comparative Analysis Of The C++, Java, And Python Languages. Keene State College^[SEP] Project from course CS430 Principles of Programming Languages.
19. Shuler, R. (2002). How does the Internet work? <https://web.stanford.edu/class/msande91si/wwwspr04/readings/week1/InternetWhitepaper.htm>
20. Talim, S. (2016). Ruby

Hashes. http://rubylearning.com/satishtalim/ruby_hashes.html The Ethics Centre (2016). Ethics, morality, law – what's the difference?. Retrieved from <https://ethics.org.au/ethics-morality-law-whats-the-difference/>

21. Urie, E. (2018). The 4 Data Structures Every New Developer Should Know. <https://learntocodewith.me/posts/data-structures/>
22. A. Ranganathan: Beyond HTML5: Database APIs and the Road to IndexedDB, Mozilla.org, <http://hacks.mozilla.org/2010/06/beyond-html5-databaseapis-and-the-road-to-indexeddb/>, accessed 5. 7. 2010
23. Microsoft Silverlight, <http://www.silverlight.net/>, accessed 5. 7. 2010
24. Web Workers, Editor's Draft, 25. 6. 2010 <http://dev.w3.org/html5/workers/>, accessed 5. 7. 2010