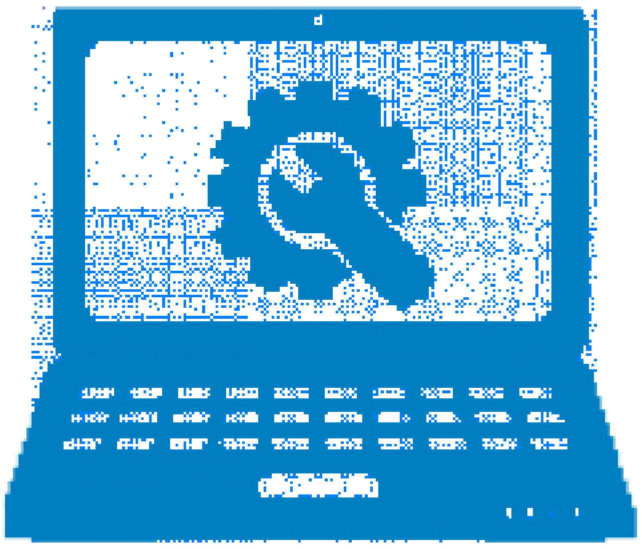
SECTION 4. SOFTWARE



LEARNING OBJETIVES

- To understand the function of the OS.
- To recognize the features of a graphical user interface (GUI).
- To use the correct determiners with countable and uncountable nouns.
- To summarise a written text.
- To understand the basic features and applications of word processors.
- To give and follow instructions.
- To understand the basic features and applications of spreadsheets and databases.
- To form and pronounce plurals correctly.

INDEX

1. Introduction

Computer **software**, or simply software, is a collection of data or computer instructions that tell the computer how to work. This is in contrast to physical hardware, from which the system is built and actually performs the work. In computer science and software engineering, computer software is all information processed by computer systems, **programs** and **data**. Computer software includes computer programs, **libraries** and related non-executable data, such as **online documentation** or **digital media**. Computer hardware and software require each other and neither can be realistically used on its own.

At the lowest programming level, executable code consists of **machine language** instructions supported by an individual processor—typically a central processing unit (CPU) or a graphics processing unit (GPU). A machine language consists of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. For example, an **instruction** may change the value stored in a particular storage location in the computer—an effect that is not directly observable to the user. An instruction may also invoke one of many input or output operations, for example displaying some text on a computer screen; causing state changes which should be visible to the user. The processor executes the instructions in the order they are provided, unless it is instructed to "jump" to a different instruction, or is interrupted by the operating system. As of 2020, most personal computers, smartphone devices and servers have processors with multiple execution units or multiple processors performing computation together, and computing has become a much more concurrent activity than in the past.

The majority of software is written in **high-level programming languages**. They are easier and more efficient for programmers because they are closer to natural languages than machine languages. High-level languages are translated into machine language using a **compiler** or an **interpreter** or a combination of the two. Software may also be written in a **low-level assembly language**, which has strong correspondence to the computer's machine language instructions and is translated into machine language using an assembler.

Software is just instructions written by a **programmer** which tells the computer what to do. Programmers are also known as 'software developers', or just plain 'developers'.

Software programs can have millions of lines of code. If one line doesn't work, the whole program could break! Even the process of starting software goes by many different names in English. Perhaps the most correct technical term is **'execute'**, as in "the man executed the computer program." Some other common verbs used to start a software program you will hear are **'run'**, **'launch**, and even **'boot'** (when the software in question is an operating system).

Software normally has both **features** and **bugs**. Hopefully more of the former than the latter! When software has a bug there are a few things that can happen. The program can crash and terminate with a confusing message. This is not good. **End users** do not like confusing error messages such as:

> ***Site error:*** *the file /home7/businfc6/public_html/blog/wordpress/wp-content/plugins/seo-blog/core.php requires the ionCube PHP Loader ioncube_loader_lin_5.2.so to be installed by the site administrator.*

Sometimes when software stops responding you are forced to manually **abort** the program yourself by pressing some strange combination of keys such as ctrl-alt-delete.

Because of poor **usability**, documentation, and strange error messages, programming still seems very mysterious to most people. That's too bad, because it can be quite fun and rewarding to write software. To succeed, you just have to take everything in small steps, think very hard, and never give up.

I think everyone studying Information Technology should learn at least one programming language and write at least one program. Why? Programming forces you to think like a computer. This can be very rewarding when dealing with a wide range of IT-related issues from tech support to setting up PPC (pay-per-click) advertising campaigns for a client's web site. Also, as an IT professional, you will be dealing with programmers on a daily basis. Having some understanding of the work they do will help you get along with them better.

There are two basic kinds of software you need to learn about as an IT professional. The first is **closed source** or **proprietary** software, which you are not free to modify and improve. An example of this kind of software is Microsoft Windows or Adobe Photoshop. This software model is so popular that some people believe it's the only model there is. But there's a whole other world of software out there.

The other kind of software is called **open source** software, which is normally free to use and modify (with some **restrictions** of course). Examples of this type of software include most popular programming languages, operating systems such as Linux, and thousands of applications such as Mozilla Firefox and Open Office.

But what is the real difference between open source and closed source software? Is open source software just about saving money? Let's investigate. Let's say for instance you find a bug in the latest version of Mozilla Firefox. The bug is causing a major project to fail and you need to fix it right away. This is not very likely to happen, I realize, but it's just an example. You might take the following steps:

Step 1. Download and unzip (or uncompress) the source code from Mozilla.

Step 2. Use an Integrated Development Environment (IDE) and a debugger to find and fix the bug in the source code. Please note that you will need to know a little C++ to debug applications such as this.

Step 3. Test the fix and then use a compiler to turn the source code into a binary file. This can take a long time for big programs. Once the source code is compiled then the program should work!

Step 4. You are almost done. Now send the bug fix back to the Mozilla Firefox team. They may even use your bug fix in the next release!

Now imagine you find a bug in a proprietary code base such as Microsoft Word. What can you do? Not much, just file a bug report and hope someone fixes it at some point.

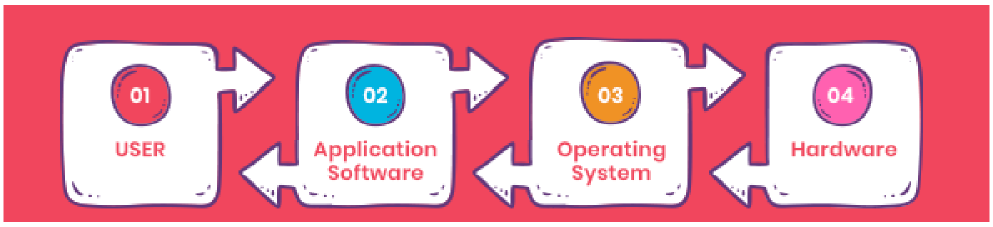
This is a rather radical example, but I think it illustrates to a large degree why programmers generally prefer open source software to closed source alternatives. Good programmers love code and they want access to it. Hiding the code from a programmer is like hiding the car engine from an auto mechanic.

2. Software types

A computer system is made up of hardware and software:

- **hardware** is the physical parts of the computer
- **software** is a general term for programs that control and make use of the hardware

Software is translated into machine code for the hardware to understand. Software types are in a hierarchy of their position in relation to the hardware.



System Software

System Software helps the user, hardware, and application software to interact and function together. These types of computer software allow an environment or platform for other software and applications to work in. This is why system software is essential in managing the whole computer system.

When you first power up your computer, it is the system software that is initially loaded into memory. Unlike application software, the System software is not used by end-users like you. It only runs in the background of your device, at the most basic level while you use other application software. This is why system software is also called **"low-level software"**.

Operating systems are an example of system software. All of your computer-like devices run on an operating system, including your desktop, laptop, smartphone, and tablet, etc. Here is a list of examples of an operating system. Let's take a look and you might spot some familiar names of system software:

For desktop computers, laptops and tablets:

- Microsoft Windows
- Mac (for Apple devices)
- Linux

For smartphones:

- Apple's iOS
- Google's Android
- Windows Phone OS

Other than operating systems, some people also classify utility software and driver software as types of system software. However, we will discuss them individually in the next two sections.

Utility Software

Utility software is considered a subgroup of System software. They manage the performance of your hardware and application software installed on your computer, to ensure they work optimally. Some features of utility software include:

- Antivirus and security software
- File compressor
- Disk cleaner
- Disk defragmentation software
- Data backup software

Driver Software

Driver software is often classified as one of the types of system software. They operate and control devices and peripherals plugged into a computer. Drivers are important because they enable the devices to perform their designated tasks. They do this by translating commands of an Operating System for the Hardware or devices, assigning duties. Therefore, each device connected with your computer requires at least one device driver to function.

Since there are thousands of types of devices, drivers make the job of your system software easier by allowing it to communicate through a standardized language. Some examples of driver software that you may be familiar with are:

- Printer Driver
- Mouse Driver
- Network Card

Usually, the operating system comes built-in with drivers for mouse, keyboard, and printers by default. They often do not require third-party installations. But for some advanced devices, you may need to install the driver externally. Moreover, if you use multiple operating systems like Linux, Windows, and Mac, then each of these supports different variants of drivers. For them, separate drivers need to be maintained for each.

Application software

It is software that uses the computer system to perform special functions or provide entertainment functions beyond the basic operation of the computer itself. There are many different types of application software, because the range of tasks that can be performed with a modern computer is so large. We will learn more about the application software in the following section.

3. Application Software

As a user of technology, Application Software or 'Apps' are what you engage with the most. These types of computer software are productive end-user programs that help you perform tasks.

Software applications are also referred to as non-essential software. They are installed and operated on a computer-based on the user's requirement. There are plenty of application software that you can use to perform different tasks. The number of such apps keeps increasing with technological advances and the evolving needs of the users. You can categorize these software types into different groups, as shown in the following table:

Application Software Type	Examples
Word processing software: Tools that are used to create word sheets and type documents etc.	Microsoft Word, WordPad, AppleWorks and Notepad
Spreadsheet software: Software used to compute quantitative data.	Apple Numbers, Microsoft Excel and Quattro Pro
Database software: Used to store data and sort information.	Oracle, MS Access and FileMaker Pro
Application Suites: A collection of related programs sold as a package.	OpenOffice, Microsoft Office
Multimedia software: Tools used for a mixture of audio, video, image and text content.	Real Player, Media Player
Communication Software: Tools that connect systems and allow text, audio, and video-based communication.	MS NetMeeting, IRC, ICQ
Internet Browsers: Used to access and view websites.	Netscape Navigator, MS Internet Explorer, and Google Chrome
Email Programs: Software used for emailing.	Microsoft Outlook, Gmail, Apple Mail

Without software applications, it would be very hard to actually perform any meaningful task on a computer unless one was a very talented, fast, and patient programmer. Applications are meant to make users more productive and get work done faster. Their goal should be flexibility, efficiency, and user-friendliness.

Today there are thousands of applications for almost every purpose, from writing letters to playing games. Producing software is no longer the lonely profession it once was, with a few random geeks hacking away in the middle of the night. Software is a big business and the development cycle goes through certain stages and versions before it is released.

Software programs are normally written and compiled for certain hardware platforms. It is very important that the software is **<u>compatible</u>** with all the components of the computer. For instance, you cannot run software written for a Windows computer on a Macintosh computer or a Linux computer. Actually, you can, but you need to have special emulation software or a virtual machine installed. Even with this special software installed, it is still normally best to run a program on the kind of computer for which it was intended.

Applications are released in different versions, including alpha versions, beta versions, release candidates, trial versions, full versions, and upgrade versions. Even an application's instructions are often included in the form of another application called a help file.

Alpha versions of software are normally not released to the public and have known bugs. They are often seen internally as a 'proof of concept'. Avoid alphas unless you are desperate or else being paid as a **'tester'**.

Beta versions, sometimes just called 'betas' for short, are a little better. It is common practice nowadays for companies to release public beta versions of software in order to get free, real-world testing and feedback. Betas are very popular and can be downloaded all over the Internet, normally for free. In general, you should be wary of beta versions, especially if program stability is important to you. There are exceptions to this rule as well. For instance, Google has a history of excellent beta versions which are more stable than most company's releases.

After the beta stage of software development comes the **release candidates** (abbreviated RC). There can be one or more of these candidates, and they are normally called RC 1, RC 2, RC 3, etc. The release candidate is very close to what will actually go out as a feature complete 'release'.

The final stage is a **'release'**. The release is the real program that you buy in a shop or download. Because of the complexity in writing PC software, it is likely that bugs will still find their way into the final release. For this reason, software companies will offer patches to fix any major problems that end users complain loudly about.

Applications are distributed in many ways today. In the past most software has been bought in stores in versions called retail boxes. More and more, software is being distributed over the Internet, as open source, shareware, freeware, or closed source versions.

Freeware

Freeware software is any software that is available to use for free. They can be downloaded and installed over the internet without any cost. Some well-known examples of freeware are:

- Google Chrome
- Skype
- Instagram
- Snapchat
- Adobe reader

Although they all fall under the category of Application or end-user software, they can further be categorized as freeware because they are free for you to use.

Shareware

Shareware, on the other hand, are software applications that are paid programs, but are made available for free for a limited period of time known as 'trial period'. You can use the software without any charges for the trial period but you will be asked to purchase it for use after the trial ends. Shareware allows you to test drive the software before you actually invest in purchasing it. Some examples of Shareware that you must be familiar with are:

- Adobe PhotoShop
- Adobe Illustrator
- Netflix App
- Matlab
- McAfee Antivirus

Open Source Software

This is a type of software that has an **open-source** code that is available to use for all users. It can be modified and shared to anyone for any purpose. Common examples of open source software used by programmers are:

- LibreOffice
- PHP
- GNU Image Manipulation Program (GIMP)

Closed Source Software

These are the types of software that are non-free for the programmers. For this software, the source code is the intellectual property of software publishers. It is also called **'proprietary software'** since only the original authors can copy, modify and share the software. Following are some of the most common examples of closed-source software:

- .Net
- Java
- Android
- Microsoft Office
- Adobe PhotoShop

In conclusion, there can be multiple ways to classify different types of computer software. The software can be categorized based on the function they perform such as Application software, System software, Programming Software, and Driver software. They can also be classified based on different features such as the nature of source code, accessibility, and cost of usage.

4. Operating Systems

The operating system is the lowest software layer that a typical user will deal with every day. That is what makes it special and worth studying in detail.

A computer would be fairly useless without an OS, so today almost all computers come with an OS pre-installed. Before 1960, every computer model would normally have it's own OS custom programmed for the specific architecture of the machine's components. Now it is common for an OS to run on many different hardware configurations.

At the heart of an OS is the **kernel**, which is the lowest level, or core, of the operating system. The kernel is responsible for all the most basic tasks of an OS such as controlling the file systems and device drivers. The only lower-level software than the kernel would be the BIOS, which isn't really a part of the operating system. We discuss the BIOS in more detail in another unit.

The most popular OS today is Microsoft Windows, which has about 85% of the market share for PCs and about 30% of the market share for servers. But there are different types of Windows OSs as well. Some common ones still in use are Windows 98, Windows 2000, Windows XP, Windows Vista, and Windows Server. Each Windows OS is optimized for different users, hardware configurations, and tasks. For instance, Windows 98 would still run on a brand-new PC you might buy today, but it's unlikely Vista would run on PC hardware originally designed to run Windows 98.

There are many more operating systems out there besides the various versions of Windows, and each one is optimized to perform some tasks better than others. Free BSD, Solaris, Linux and Mac OS X are some good examples of non-Windows operating systems.

These computer platforms differ in areas such as device installation, network connectivity or compatibility with application software.

Geeks often install and run more than one OS on a single computer. This is possible with dual-booting or by using a virtual machine. Why? The reasons for this are varied and may include preferring one OS for programming, and another OS for music production, gaming, or accounting work.

The function of the OS

An operating system is a generic term for the **multitasking** software layer that lets you perform a wide array of **'lower level tasks'** with your computer. By low-level tasks we mean:

- the ability to sign in with a username and password
- sign out the system and switch users
- format storage devices and set default levels of file compression
- install and upgrade device drivers for new hardware
- install and launch applications such as word processors, games, etc
- set file permissions and hidden files
- terminate misbehaving applications

GUI operating systems

The term user interface refers to the standard procedures that the user follows in order to interact with a computer. In the late 1970s and early 80s, the way users accessed computer systems was very complex. They had to memorize and type a lot of commands just to see the contents of a disk, to copy files or to respond to a single prompt. In fact, it was only experts who used computers, so there was no need for a user-friendly interface.

An OS must have at least one kind of **user interface**. Today there are two major kinds of user interfaces in use, the **command line interface (CLI)** and the **graphical user interface (GUI)**. Right now, you are most likely using a GUI interface, but your system probably also contains a command line interface as well.

Typically speaking, GUIs are intended for general use and CLIs are intended for use by computer engineers and system administrators. Although some engineers only use GUIs and some diehard geeks still use a CLI even to type an email or a letter.

In 1984, Apple produced the Macintosh, the first computer with a mouse and a graphical user interface (GUI). Macs were designed with one clear aim: to facilitate interaction with the computer. A few years later, Microsoft launched Windows, another operating system based on graphics and intuitive tools. Nowadays, computers are used by all kinds of people, and as a result there is a growing emphasis on accessibility and **user-friendly** systems.

A GUI makes use of a **WIMP** environment: windows, icons, menus and pointer. The background of the screen is called the desktop, which contains labelled pictures called icons. These icons represent files or folders. Double-clicking a folder opens a window which contains programs, documents, or more nested folders. When you are in a folder, you can launch a program or document by double-clicking the icon, or you can drag it to another location. When you run a program, your PC opens a window that lets you work with different tools. All the programs have a high level of consistency, with similar toolbars, menu bars, buttons and dialog boxes. A modern OS also provides access to networks and allows multitasking, which means you can run several programs — and do various tasks - at the same time.

Examples of popular operating systems with GUI interfaces include Windows and Mac OS X. Unix systems have two popular GUIs as well, known as KDE and Gnome, which run on top of X-Windows. All three of the above mentioned operating systems also have built-in CLI interfaces as well for power users and software engineers. The CLI in Windows is known as MS-DOS. The CLI in Max OS X is known as the Terminal. There are many CLIs for Unix and Linux operating systems, but the most popular one is called Bash.

In recent years, more and more features are being included in the basic GUI OS install, including notepads, sound recorders, and even web browsers and games. This is another example of the concept of 'convergence' which we like to mention.

A great example of an up and coming OS is Ubuntu. Ubuntu is a Linux operating system which is totally free, and ships with nearly every application you will ever need already installed. Even a professional quality office suite is included by default. What's more, thousands of free, **ready-to-use** applications can be downloaded and installed with a few clicks of the mouse. This is a revolutionary feature in an OS and can save lots of time, not to mention hundreds or even thousands of dollars on a single PC. Not surprisingly, Ubuntu's OS market share is growing very quickly around the world.

As an IT professional, you will probably have to learn and master several, if not all, the popular operating systems. If you think this sort of thing is fun and interesting, then you have definitely chosen the right career ;)

5. Language work

Countable and uncountable nouns

- **Countable nouns** are people or things that we can count. They have a singular and a plural form (e.g. file, program, system, application)
- **Uncountable nouns** are things that we can't count. They have no plural form (e.g. software, music, robotics, multimedia, networking, storage).
    - *A lot of software these days is open-source.*
    - *Not: A lot of softwares these days are open-source*
- Some words are countable in many languages but uncountable in English, and are used with a singular verb (e.g. advice, damage, equipment, furniture, research, news, progress, homework).
    - *The advice he gave me was very useful*
- **Countable** nouns must have a **determiner** (a, the, my, this, etc.) in the **singular**, although this is not necessary in the plural.
    - *I deleted the file yesterday.*
    - *I lost more than 300 files when my computer crashed.*
- We use a before **a** consonant sound and **an** before a vowel. The definite article the means you know which one/ones mean.
    - *An icon is a small graphic.*
    - *The icons on the toolbar are used to ..*
- We don't use a/an with uncountable nouns.
    - *Not: a robotics*
- We don’t use the in generalizations with uncountable nouns of plural countable nouns.
    - *I like music.*
    - *Not: I like the music.*
    - *Computer programs are expensive*
    - *Not: The computer programs are expensive.*
- Countable and uncountable nouns take different determiners.
    - **Many, few, a few** only go with countable nouns.
        - *There are many versions of Windows Vista.*
    - **Much, little, a little, a great deal of** only go with uncountable nouns.
        - *I have a little time free this afternoon if you want to meet.*

Plurals

- In most cases, we form the plural in English by adding -s.

  record → *records*

- If a word ends in -s, -sh, -x or -ch, we add -es.

  address →*addresses*

  index→ *indexes*

- If a word ends in a consonant + y, the y becomes i and we add -es.

  company → *companies*

  facility →*facilities*

- However, if the y follows a vowel, we add only -s.

  birthday → *birthdays*

- There are several irregular plural forms:

  man/woman →*men/women*

  child → *children*

  analysis → *analyses*

  formula→*formulae (or formulas)*

  criterion →*criteria*

  Mouse →*mice*

- The -s is pronounced as:
  - /s/ after one of these sounds: /p/, /t/, /k/, /f/ of /θ/ (e.g. amounts, hyperlinks)
  - /iz/ after one of these sounds: /s/, /z/, /ʃ/ , /tʃ/ or /dʒ/ (e.g. businesses, devices, mages)
  - /z/ in most other cases (e.g. files, fields, customers, columns)

Giving instructions

- To give instructions, we use the imperative form of the verb and sequence words such as first, next, then, after that, finally, etc.

  First*, use the mouse to select the text.*
  Then *choose the Cut command from the Edit menu.*
  Next*, choose Paste from the Edit menu.*
  Finally*, check that the text has appeared in the right place.*

- We can also use the present simple with **you**.

  Now ***you find*** *where you want the text to appear and* ***you click*** *to position the insertion point.*

Following instructions

- I you want to check that you have understood instructions, you can use expressions like

  Like this?
  Is that right?

- If you want to signal that you are ready to move on to the next step, you can use expressions like:

  OK, I've done that now.
  What next?

- If you want to ask if the process is completed, you can use expressions like:

  Is that everything?
  Anything else?

6. SELF-ASSESMENT

1) **'crash'**
   a) an error in a computer program
   b) something a computer program is "supposed" to do; these are often reasons to use a particular program or upgrade to a more recent version
   c) a computer failure due to faulty hardware or a serious software bug
2) **'compatible'**
   a) long-term, persistent, does not require power to retain it's state
   b) capable of being used without modification
   c) unable to be changed
3) **'IDE' or 'integrated development environment'**
   a) an application normally consisting of a source code editor, a compiler and/or interpreter, build-automation tools, and a debugger
   b) software in which the license stipulates that the user cannot see, edit, or manipulate the source code of a software program
   c) a rule or law which limits or controls something
4) **'end user'**
   a) a person who uses a product or service on a computer
   b) a position responsible for on-demand support for end users including: fixing hardware, installing software, and troubleshooting minor network issues
   c) a person who writes or modifies computer programs or applications
5) **'closed source'**
   a) a measure of how easy or efficient a program is to use
   b) software in which the license stipulates that the user cannot see, edit, or manipulate the source code of a software program
   c) an incorrect action attributable to poor judgment, ignorance, or inattention
6) **'restriction'**
   a) an error in a computer program
   b) software in which the license stipulates that the user cannot see, edit, or manipulate the source code of a software program
   c) a rule or law which limits or controls something

7) **'proprietary'**
   a) a computer failure due to faulty hardware or a serious software bug
   b) software in which the license stipulates that the user cannot see, edit, or manipulate the source code of a software program
   c) privately developed and owned technology

8) **'usability'**
   a) a measure of how easy or efficient a program is to use
   b) an error in a computer program
   c) a program in which the code is distributed allowing programmers to alter and change the original software as much as they like

9) **'programmer'**
   a) a person who uses a product or service on a computer
   b) a person who writes or modifies computer programs or applications
   c) a position responsible for making complex data structures easy to understand and navigate; especially critical at the beginning of new software development projects to ensure the application performs in a useful way for it's intended end-users

10) **'bug'**
   a) an error in a computer program
   b) a rule or law which limits or controls something
   c) a measure of how easy or efficient a program is to use

11) **'open source'**
   a) privately developed and owned technology
   b) an application normally consisting of a source code editor, a compiler and/or interpreter, build-automation tools, and a debugger
   c) a program in which the code is distributed allowing programmers to alter and change the original software as much as they like

12) **'feature'**
   a) an incorrect action attributable to poor judgment, ignorance, or inattention
   b) a computer failure due to faulty hardware or a serious software bug
   c) something a computer program is "supposed" to do; these are often reasons to use a particular program or upgrade to a more recent version

13) 'OS' or 'operating system'

a) concurrent execution of two or more tasks by a processor

b) a file which does not appear by default in a directory listing; normally for security reasons or to spare confusion in end users

c) a GUI or CLI software link between the computer and operator; also provides a framework for productivity software such as an office suite, web browser, or programming languages

14) 'VM' or 'virtual machine'

a) a software program which mimics the performance of one or more hardware devices in order to run software independently of the actual hardware

b) a set of strict rules for controlling read, write, and execute access to a file or directory

c) a file which does not appear by default in a directory listing; normally for security reasons or to spare confusion in end users

15) 'GUI' or 'graphical user interface'

a) a file which does not appear by default in a directory listing; normally for security reasons or to spare confusion in end users

b) a text-only link between a computer and its operator

c) an icon based link between a computer and its operator

16) 'multitasking'

a) a text-only link between a computer and its operator

b) a file which does not appear by default in a directory listing; normally for security reasons or to spare confusion in end users

c) concurrent execution of two or more tasks by a processor

17) 'X' or 'X Window System'

a) an open source version of Unix developed by a volunteer team of programmers around the world

b) a popular open source office suite initiated by Sun Microsystems designed to compete specifically with MS Office.

c) a software toolkit for UNIX systems underlying numerous GUI window managers including KDE and Gnome

18) 'device driver'

a) prepare a device to store data, erasing any existing data

b) the fundamental part of an operating system responsible for providing access to the machine's hardware

c) software which converts the data from a component or peripheral into data that an operating system can use

19) 'CLI' or 'command line interface'

a) a text-only link between a computer and its operator

b) prepare a device to store data, erasing any existing data

c) an icon based link between a computer and its operator

20) 'Linux'

a) an open source version of Unix developed by a volunteer team of programmers around the world

b) a software toolkit for UNIX systems underlying numerous GUI window managers including KDE and Gnome

c) an 8-bit binary code used to represent standard American English numbers and letters

ANSWERS

1	2	3	4	5	6	7	8	9	10
C	B	A	A	B	C	B	A	B	A
11	12	13	14	15	16	17	18	19	20
C	C	C	A	C	C	C	C	A	A

7. BIBLIOGRAPHY

- https://edu.gcfglobal.org
- https://www.english4it.com
- https://es.wikipedia.org/
- https://dictionary.cambridge.org/es/gramatica
- https://learnenglish.britishcouncil.org/english-grammar-reference/
- https://www.bbc.co.uk/bitesize/guides/z6r86sg/revision/5
- https://en.wikipedia.org
- https://www.goodcore.co.uk/blog/types-of-software/
- *Santiago Remacha Esteras. Infotech, English for computers users Student's Book.*