**ISD lab sheet week 1**

Create your logbook document, based on the template provided.

Start your logbook by answering the following questions:

Answer the following questions (in writing) to reflect on the contents of the lecture today. You can use the lecture slides as well as any source you find online, however, if you use an online source, please briefly mention it in your answer (like “as seen on <https://www.python.org/>”).

Elaborate in your answers where you are asked to explain something. Your answers will be part of your logbook (once we have set it up) and therefore please store your answer document (preferable .doc or .docx) on your UWL cloud storage (see lecture slides) or wherever convenient for you.

Your answer document should be at least a page long. You can write more but also keep your answers concise.

Questions:

1. What is a code repository (often also called version control system) used for?

A code repository is a file archive and web hosting platform where huge amounts of source code is kept for either software or web pages. They are in a publicly or privately platform. They are mostly used by open-source software projects and other developer projects such as SDK (software development kits) or git repository. There is a variety platform that can be used to host the software project they range from google cloud, Microsoft azure, oracle cloud, amazon web service and GitHub on these platforms they are stored on a cloud server.

1. Why is it advantageous to use a code repository? There are many reasons why code repository are handy this is because it is easier to compile code on a platform and you can work with other developer on your project to code your application you want. You can also have the advantage of getting help from the community of developers so ther=y can help solve problems within your code.
2. Describe the different “layers” of Software that exist on a typical computer and explain why there are different layers of software.

There is four layers of things in typical computer they are hardware abstraction layer, operating system layer system service layer and the application layer. The hardware abstraction layer is where the bootloader, board support package, device drivers, and other components this is there to load the software on to the operation system. The OS layer is a software system for uniformly managing hardware resources. It abstracts many hardware functions and provides them to applications in the form of services. Scheduling, files synchronization, and networking are the most common services provided by the OS. The OS are widely used in most desktops and embedded systems. The common embedded OS contains embedded Linux, Windows CE, VxWorks, MeeGo, Tizen, Android, Ubuntu, and some operating systems used in specific fields. The system service layer is the service interface that the OS provides to the application. Using this interface, applications can access various services provided by the OS. To some extent, it plays the role of a link between the OS and applications. The application, located at the top level of the software hierarchy, implements the system functionality and business logic. From a functional perspective, all levels of modules in the application aim to perform system functions. From a system perspective, each application is a separate OS process. Typically, applications run in the less-privileged processor mode and use the API system schedule provided by the OS to interact with the OS.

1. Describe what an algorithm is and explain why it is a useful “tool” to translate from a human level problem (we can think of) to a computer program.

In mathematics and computer science, an algorithm is an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing and automated reasoning tasks. As an effective method, an algorithm can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.