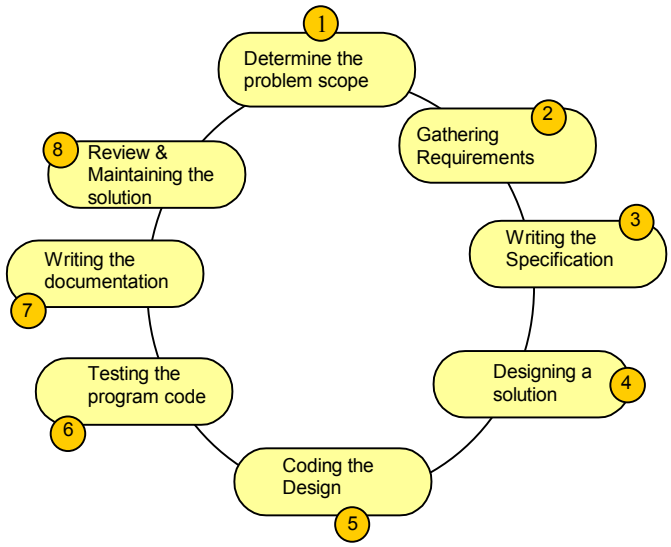
Software Design

Software development Life Cycle

Program development forms part of a natural life cycle. New solutions are designed and implemented because:

-an older system is failing
-user needs have changed.

In order to design and implement a new solution it is important to put this activity into a wider context, to see what part of the overall process it represents.
Each stage of this cycle is listed below in more detail.



Stage 1: Determining the problem scope
Scope is a term used to describe the **boundaries** of the problem (often called the `**problem domain').** In particular, it is a **definition** of **what is** and **what is not** covered by the identified problem. It also includes those who are affected by the problem (e.g. people, organisations etc.) and how they are affected
Understanding the scope is vital in order to understand the nature of the problem.

Stage 2: Gathering requirements
Before the problem is tackled, it is vital that the **functional requirements -** what is **wanted** from the solution by **those who are affected -** are clearly known. Those affected by the problem are commonly referred to as stakeholders. This will absolutely specify what the solution **does** and **does not** cover. A staggeringly high number of commercial solutions fail because these requirements are poorly researched; it is a common failure of IT projects worldwide. Functional requirements may be collected in the form of **use cases;** a use case is simply a way of **describing how** a single task is achieved.

Stage 3: Writing the specification
This is a itemised list of the elements required in order to start designing a solution: • inputs • outputs • processing • storage • user interface • constraints

Stage 4: Designing a solution

Using a suitable design tool or methodology (a set of procedures or methods) a solution is built. This may be paper based or electronic depending on the design tools being used.

Modern IT solutions can be partly solved by software tools; programs effectively writing new solutions. Avoid the temptation to rush ahead into the coding; many mistakes are made this way and a lot of time needlessly wasted! A number of different design tools are examined later.

Stage 5 Coding the design

From here, designs are converted into program code by experienced software developers. The programming language will have been selected before they start as some design features will lend themselves to particular languages.
It is common for software developers to work in a team. If this is the case, it is important that they are all clear about their **individual responsibilities** and adopt similar **working practices** and the **`in-house' style in an effort to standardise.**
Programs written in a **modular** way (broken into **smaller modules** such as **procedures** and **functions) ensure reusability** through **shared libraries.** Use of **classes** will also permit easy code reuse and speed up the development process.
Ideally, all developers need to have an understanding of how their modules fit into the larger solution, even if they **do not know** how modules which they have not written **actually** work.

Stage 6: Testing the program code

This is vital.
Two common testing methods are used: black box and white box.

Black box testing does not care about how the program was written (i.e. peeking inside). It only wants to see how closely a program meets its list of functional requirements: Does it do what it is supposed to do?

White box testing examines the performance of the program code, ensuring that what has been programmed is generating the right results. White box testing takes much more time and usually starts after black box testing has been completed.

Stage 7: Writing documentation

Documentation is **also** vital!
Generally there will be three types of documentation:

- **internal documentation**
- **technical documentation**
- **user documentation**.

Internal documentation is documentation that is actually inside the program's code.
Internal documentation is documentation that is actually inside the program's code.
All good programs should be self-documenting; there are some easy ways to achieve this:

- **meaningful** and **sensible** variable names
- good use of **comments,** written to describe the **code's purpose in the solution, not** the syntax of the actual code itself
- good **indentation** and tidy **layout** (most modern IDEs do this automatically for the programmer).

See later **technical documentation and user documentation** in greater detail.

Stage 8: Reviewing and maintaining the solution
This stage's components can be examined separately. Review is a reflective process, **looking** at the solution and **comparing** it to the **stakeholders' original requirements. Have** they been met? And if so, **how well** is the solution working? Maintenance is an ongoing process. Programs are written to solve a problem which exists at a particular)int in time; as business or personal needs change, the solution will seem less ideal. Maintenance is the process of **keeping** the solution **working, fixing** minor errors that occur, making ,mall **adjustments** which **expand** the **scope** of the program.
If the maintenance requirements become too severe, it may be necessary to redevelop - to start the Cycle again.

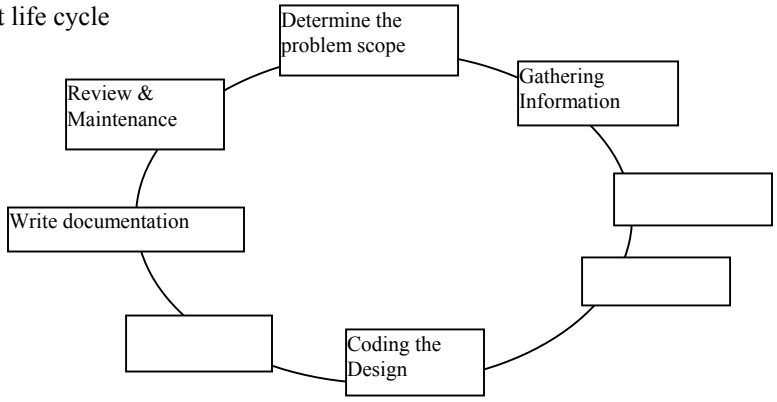
Software Design

Name :

1) In program development, new programming solutions are implemented because

_____ or _____

2) Complete the eight stages of this development life cycle



3) Many mistakes are made and time wasted during the Design stage because ……

4) Name three things which a team of programmers must ……..

5) Name two common types of testing used in software development.

6) Name three types of documentation associated with a computer system.