Q1. What are comments and what is the importance if commenting in any code?

Ans: A comment is a programmer-readable explanation or *annotation* in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters. The syntax of comments in various programming languages varies considerably.

Comments are sometimes also processed in various ways to generate documentation external to the source code itself by documentation generators, or used for integration with source code management systems and other kinds of external programming tools.

The flexibility provided by comments allows for a wide degree of variability, but formal conventions for their use are commonly part of programming style guides.

Comments are important in any code because of the following two reasons:

- In an organization, we work in a team; there are many programmers who work on the same project. So, the well commented functions/logics are helpful to other programmers to understand the code better. They can easily understand the logic behind solving any problem.

- If you see/edit code later, comments may help you to memorize your logic that you have written while writing that code.

  Sometimes, it happens with lazy programmers (who do not comment the code properly) that they forget their implemented logics and waste much more time solving the issue.

  So, I would recommend you please comment the code properly so that you or your colleagues can understand the logic better. Writing comments may take time, but it maintains the international coding standards.

Q2. What is Call Statement and when do you use this statement?

Ans: The call statement transfers control from one object program to another within the run unit.

The program containing the call statement is the calling program; the program identified in the call statement is the called subprogram. Called programs can contain call statements; however, only programs defined with the recursive clause can execute a call statement that directly or indirectly calls itself.

Use the call statement to transfer program control from the calling program to an external subroutine or program that has been compiled and cataloged.

The first time a call is executed, the system searches for the subroutine in a cataloged library and changes a variable that contains the subroutine name to contain its location information instead. This procedure eliminates the need to search the catalog again if the same subroutine is called later in the program. For indirect calls, the variable specified in the call as the @variable is used; for direct calls, an internal variable is used. With the indirect method, it is best to assign the subroutine name to the variable only once in the program, not every time the indirect call statement is used.

Arguments are variables, arrays, array variables, expressions, or constants that represent actual values. You can pass one or more arguments from the calling program to a subroutine. The number of arguments passed in a call statement must equal the number of arguments specified in the subroutine statement that identifies the subroutine. If multiple arguments are passed, they must be separated by commas. If an argument requires more than one physical line, use a comma at the end of the line to indicate that the list continues.

If argument is an array, it must be preceded by the mat keyword, and the array should be named and dimensioned in both the calling program and the subroutine before using this statement. If the array is not dimensioned in the subroutine, it must be declared using the mat keyword in the subroutine statement. Other arguments can be passed at the same time regardless of the size of the array.

The actual values of arguments are not passed to the subroutine. Instead, a pointer to the location of each argument is passed. Passing a pointer instead of the values is more efficient when many values need to be passed to the subroutine. This method of passing arguments is called passing by reference; passing actual values is called passing by value.

Q3. How do you compile a code in VBA? What are some of the problem that you might face when you don't compile a code?

Ans: VBA checks each line as you're typing the code and highlights the syntax error as soon as the line is incorrect and you hit enter. Compile errors, on the other hand, are only identified when the entire code is analyzed by VBA.

Below are some scenarios where you'll encounter the compile error:

1. Using an IF Statement without the End IF
2. Using For statement with the Next
3. Using Select statement without using the End Select
4. Not declaring the variable (this works only when Option Explicit is enabled)

5. Calling a Sub/Function that does not exist (or with wrong parameters)

The problem that you might face when you don't compile a code:

1. Syntax errors – A specific line of code is not written correctly

2. Compile errors – Issues that happen when putting together lines of code, though the individual lines of code seem to make sense

3. Runtime errors – When the code is usually correct in principle, but an action taken by the user or the data being used leads to unexpected errors.

Q4. What are hot keys in VBA? How can you create your own hot keys?

Ans: 1 – Alt+F11 to Open the VB Editor

2 – Store Your Macros in The Personal Macro Workbook

3 – Ctrl+Space to Auto Complete

4 – Intellisense for Worksheets

6 – F8 to Step Through Each Line of Code

8 – Automate Repetitive Tasks with The For Next Loop

9 – Use Option Explicit

10 – Excel Tables (ListObjects)

11 – Get Code with the Macro Recorder

12 – The Immediate Window

14 – Check That a Range is Selected

15 – Ctrl + Y to Delete a Line of Code

16 – Ctrl + i for Quick Info

17 – Ctrl+J Opens the Intellisense Drop Down

18 – Worksheet Functions

Assign Macros to Shapes

To assign a macro so a shape:

1. Insert a shape on a worksheet, and format it to your liking. This will usually be a rectangular or circle shape that contains text.
2. Right-click the shape and choose “Assign Macro…”.

3. Select the macro from the list and press OK. The macro will usually be one that is stored in the same workbook as the shape.
4. Click off the shape by selecting a cell in the worksheet.
5. When you hover the shape the cursor will change to the hand pointer. Clicking the shape will run the macro.

Assign a Keyboard Shortcut to a Macro

To assign the keyboard shortcut:

1. Press the Macros button on the Developer or View Tab in the ribbon.
2. Select the file that contains the macro from the Macros In drop down.
3. Select the macro from the list box.
4. Press the “Options…” button.
5. Type the letter in the Shortcut key box that you want to assign the macro to. All shortcuts will start with Ctrl. You can hold the Shift key while typing the letter to create a Ctrl+Shift shortcut. This is usually recommended because most of the Ctrl+key combinations already have dedicated keyboard shortcuts in Excel.
6. Press OK and close the macros window.
7. You can now press the shortcut key combination to run the assigned macro.

Q6. What are the shortcut keys used to

a. Run the code

b. Step into the code

c. Step out of code

d. Reset the code

Ans: a. Ctrl+' and Ctrl+Enter

b. Alt+Shift+F7

c. Shift+F11

d. Press Ctrl + Shift + P