Q1. What are the fprintf and fscanf function? Write down the general form.

→ The functions fprintf and fscanf perform Input output operations that are familar to the fprint and scanf functions. The functions fprintf and fscanf are work on files. The first argument of these functions is a file pointer. The general form of ~~fprin~~ fprintf is

```
fprintf (fp, "control string", list);
```

where fp is a file pointer that has been opened for writing. The control string contains output for the item in the list. The list may include variables, constants and strings.
For example,

```
fprintf (f1, "%s %d %f", name, age, 7.5);
```

Here, name is an array variable of type char and age is an int variable.
The general format of fscanf is

```
fprintf (fp, "control string", list);
```

This statement would cause the reading of the items in the list from the file fp.
For example,

```
fscanf (f2, "%s %d", item, &quantity);
```

Sub :

Day

Time : Date : / /

Some operations of different function's :

Function Name	Operation
fopen()	Creates a new file for use. Opens an existing file for use.
fclose()	Closes a file which has been opened for use
getc()	Reads a character from a file.
putc()	Writes a character to a file.
fprintf()	Writes a set of data values to a file.
fscanf()	Reads a set of data values to a file.
getw()	Reads an integer from a file.
putw()	Writes an integer to a file.
fseek()	sets the position to a desired point in the file.
ftell()	Gives the current position in the file.
rewind()	Sets the position to the begining of the file.

Sub : Day Time : Date : / /

Mode of File Management :

→ Mode specifies the purpose of opening a file.

some mode of file management :

1. r : This mode open the file for reading only. statement :

```
FILE *P1;
P1 = fopen ("data","r");
```

2. W: This mode open the file for writing only. statement :

```
FILE *P2;
P2 = fopen ("results","w");
```

3. a: This mode open the file for adding data to it. statement :

```
FILE *P3;
P3 = fopen ("shedule","a");
```

Extra

4. r+ : This mode open the file for both opening and writing.

5. W+ : This mode open the file for both opening and writing.

6. a+ : This mode open the file for both opening and writing.

Sub: Day Time: Date: / /

Q. How to store data in a file in the secondary memory?

→ If we want to store data in a file in the secondary memory, we must specify certain things about the file, to the operating system. They include the following:

1. File Name
2. Data structure
3. Purpose.

File name is a string of characters that make up a valid filename for the operating system. It may certain two parts, a primary name and an optional period with the extention.

Example: Input.data
Student.c

Data structure of a file is defined as FILE in the library of standard input-output function definitions. Therefore, all file should be declared as type FILE before they used. FILE is a defined data type.

When we open a file, we must specify what we want to do with the file. For example, we may write data to the file or read the already existing data.

Sub: _____ Day | Time: Date: / /

Following is the general format for declaring and opening a file

```
FILE *fp;
fp = fopen("filename", "mode");
```

The first statement declares that the variable fp as a pointer to the data type FILE. And we know FILE is a structure that is defined in the input-output library. The second statement opens the file named filename and assigned as identifier to the FILE type pointer fp.
The second statement also specifies the purpose of opening this file. Mode can does this job.
Mode can be one of the following:

r → open the file for reading only.
w → open the file for writing only.
a → open the file for appending (or adding) data.

Note that the file name and mode are specified as strings. They should be enclosed in double quotation marks.

Sub: _____ Day | | | | | | | |

Time: Date: / /

Error handling within input or output operation

→ It is possible that an error may occur during input-output operations on a file. Typical error situations include to following:

1. Trying to read beyond the end of file mark.
2. Device overflow.
3. Trying to use a file that has not been opened.
4. Trying to perform an operation on a file when the file is opened for another type of operation.
5. Opening a file with an invalid file name.
6. Attempting to write to a write protected file.

If we fail to check such read and write errors, a program may behave abnormally when an error occurs. An unchecked error may result in a premature termination of the program or incorrect output. Fortunately, we have two status-inquiry library functions, "feof" and "ferror" that can help us to detect Input-output errors in the files.

The "feof" function can be used to test for an end of file condition.

Sub :

Day

Time : Date : / /

If fp is a pointer to the file that has just been opened for reading. Then the statement is:

```
if (feof(fp))
    printf ("End of data.\n");
```

Would display the message "End of data". on reaching the end of file condition.

The ferror function reports the status of the file indicated. It also takes a FILE pointer as its argument and return a non-zero integer if an error has been detected up to that point, during processing. It returns 0, otherwise. The statement

```
if (ferror(fp) != 0)
    printf ("An error has occured.\n");
```

would print the error message, if the reading is not successful.

We know that whatever a file is opened using fopen function, a file pointer is returned. If the file can not be opened for some reason, then the function returns a Null pointer.

```
if (fp == NULL)
    printf ("File could not be opened.\n");
```