

DAILY ASSESSMENT FORMAT

Date:	20 th June 2020	Name:	Sushmitha R Naik
Course:	Solo learning	USN:	4AL16EC090
Topic:	<ul style="list-style-type: none"> Files and Error Handling Pre-Processor Certificate 	Semester & Section:	6 th B
GitHub Repository:	Sushmitha_naik		

FORENOON SESSION DETAILS

The screenshot shows a web browser window displaying the SoloLearn website. The page title is "The Preprocessor Module 8 Quiz". The user's profile name "SUSHMITHA NAIK" is visible in the top right corner. The quiz progress bar shows 1/5 questions completed. The question text is: "Fill in the blanks to define a macro that triples its argument. Then use the num variable as its argument and output the result." The code snippet provided is:

```
#define TRIPLE(x) (x) * 3
int num;
printf("%d", TRIPLE(4));
```

 A green "Correct!" message with a checkmark and "21 COMMENTS" is displayed in the center. The bottom of the browser window shows the Windows taskbar with various application icons and the system clock indicating 17:14 on 20-06-2020.

The screenshot shows the SoloLearn website with the quiz "Files & Error Handling Using Error Codes". The user's profile name "SUSHMITHA NAIK" is visible in the top right corner. The quiz progress bar shows 4/4 questions completed. The question text is: "Fill in the blanks to open the file for reading and check if the end of file if reached." The code snippet provided is:

```
FILE* fptr = fopen("test", "r");
if (feof(fptr)) {
    printf("End of file reached\n");
}
```

 A green "Correct!" message with a checkmark and "15 COMMENTS" is displayed in the center. The bottom of the browser window shows the Windows taskbar with various application icons and the system clock indicating 17:14 on 20-06-2020.

Report –

Accessing Files:

- An external file can be opened, read from, and written to in a C program. For these operations, C includes the FILE type for defining a file stream. The file stream keeps track of where reading and writing last occurred.
- The stdio.h library includes file handling functions:
- FILE Typedef for defining a file pointer.
- fopen(filename, mode) Returns a FILE pointer to file filename which is opened using mode. If a file cannot be opened, NULL is returned.

Mode options are:

- r open for reading (file must exist)
- w open for writing (file need not exist)
- a open for append (file need not exist)
- r+ open for reading and writing from beginning
- w+ open for reading and writing, overwriting file
- a+ open for reading and writing, appending to file

fclose(fp) Closes file opened with FILE fp, returning 0 if close was successful. EOF (end of file) is returned if there is an error in closing.

```
#include <stdio.h>
```

```
int main () {  
FILE *fptr;  
fptr = fopen ("myfile.txt",  
"w"); if (fptr == NULL) {  
printf ("Error opening  
file."); return -1;  
}  
fclose(fptr);  
return 0;  
}
```

Reading from a File:

- The stdio.h library also includes functions for reading from an open file. A file can be read one character at a time or an entire string can be read into a character buffer, which is typically a char array used for temporary storage.
- fgetc(fp) Returns the next character from the file pointed to by fp. If the end of the file has been reached, then EOF is returned.
- fgets(buff, n, fp) Reads n-1 characters from the file pointed to by fp and stores the string in buff. A NULL character '\0' is appended as the last character in buff. If fgets encounters a newline character or the end of file before n-1 characters is reached, then only the characters up to that point are stored in buff.
- fscanf(fp, conversion_specifiers, vars) Reads characters from the file pointed to by fp and assigns input to a list of variable pointers vars using conversion_specifiers. As with scanf, fscanf stops reading a string when a space or newline is encountered.

Pre-processor Directives:

The C preprocessor uses the # directives to make substitutions in program source code before compilation.

For example, the line `#include <stdio.h>` is replaced by the contents of the `stdio.h` header file before a program is compiled.

Preprocessor directives and their uses:

#include Including header files.

#define, #undef Defining and undefining macros.

#ifdef, #ifndef, #if, #else, #elif, #endif Conditional compilation.

#pragma Implementation and compiler specific.

#error, #warning Output an error or warning message An error halts compilation.

Certificate:






