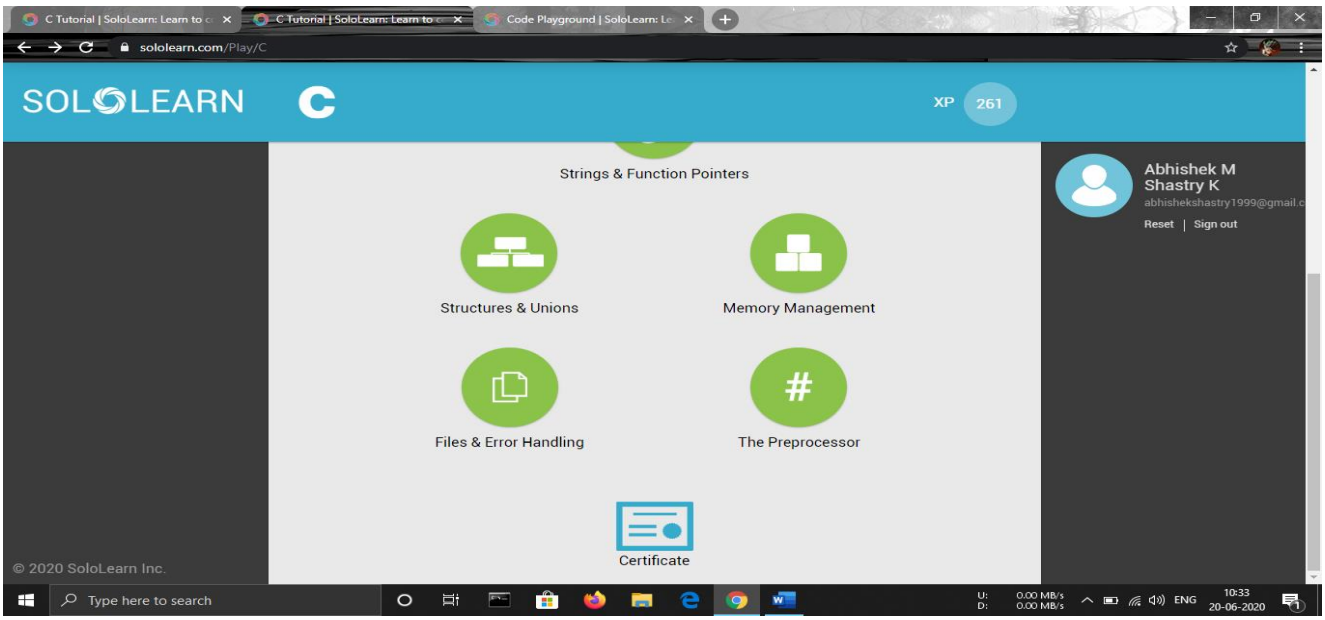
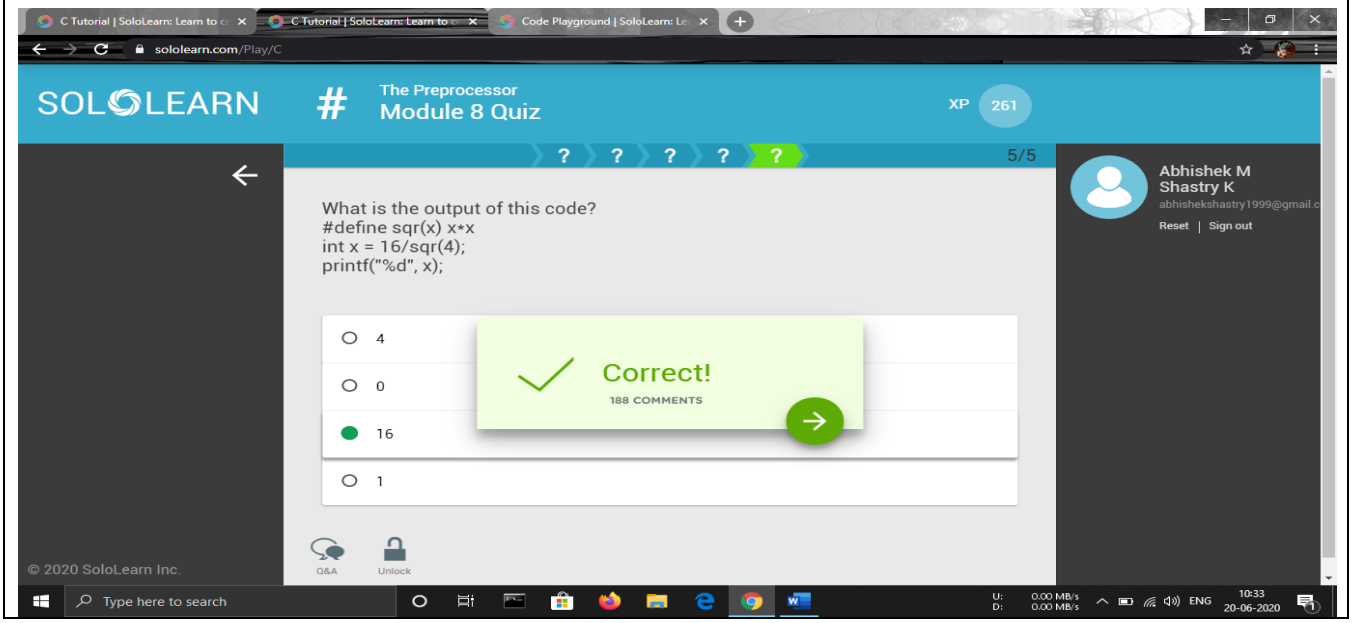
DAILY ASSESSMENT REPORT

Date:	**20/06/2020**	**Name:**	**Abhishek M Shastry K**
Course:	**C Tutorial by SOLOLEARN**	**USN:**	**4AL17EC002**
Topic:	**1] Files & Error Handling** **2] The Preprocessor**	**Semester & Section:**	**6th 'A'**
Github Repository:	**AbhishekShastry-Courses**		

FORENOON SESSION DETAILS

Image of session





Report

Files & Error Handling

- **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.
- **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.

- **Writing to a File** - The stdio.h library also includes functions for writing to a file. When writing to a file, newline characters '\n' must be explicitly added.
- **fputc**(char, fp) Writes character char to the file pointed to by fp.
- **fputs**(str, fp) Writes string str to the file pointed to by fp.
- **fprintf**(fp, str, vars) Prints string str to the file pointed to by fp. str can optionally include format specifiers and a list of variables vars.
- **Exception Handling** - Central to good programming practices is using error handling techniques. Even the most solid coding skills may not keep a program from crashing should you forget to include exception handling. An exception is any situation that causes your program to stop normal execution. Exception handling, also called error handling, is an approach to processing runtime errors.
- Some library functions, such as **fopen()**, set an error code when they do not execute as expected. The error code is set in a global variable named **errno**, which is defined in the **errno.h** header file. When using **errno** you should set it to 0 before calling a library function.
- To output the error code stored in **errno**, you use fprintf to print to the stderr file stream, the standard error output to the screen. Using **stderr** is a matter of convention and a good programming practice.
- You can output the **errno** through other means, but it will be easier to keep track of your exception handling if you only use **stderr** for error messages.
- When a library function sets **errno**, a cryptic error number is assigned. For a more descriptive message about the error, you can use **perror()**. You can also obtain the message using **strerror()** in the string.h header file, which returns a pointer to the message text.
- Some of the mathematical functions in the **math.h** library set errno to the defined macro value **EDOM** when a domain is out of range. Similarly, the **ERANGE** macro value is used when there is a range error.
- In addition to checking for a NULL file pointer and using **errno**, the **feof()** and **ferror()** functions can be used for determining file I/O errors:
    - ✓ **feof**(fp) Returns a nonzero value if the end of stream has been reached, 0 otherwise. feof also sets EOF.
    - ✓ **ferror**(fp) Returns a nonzero value if there is an error, 0 for no error.

The Preprocessor

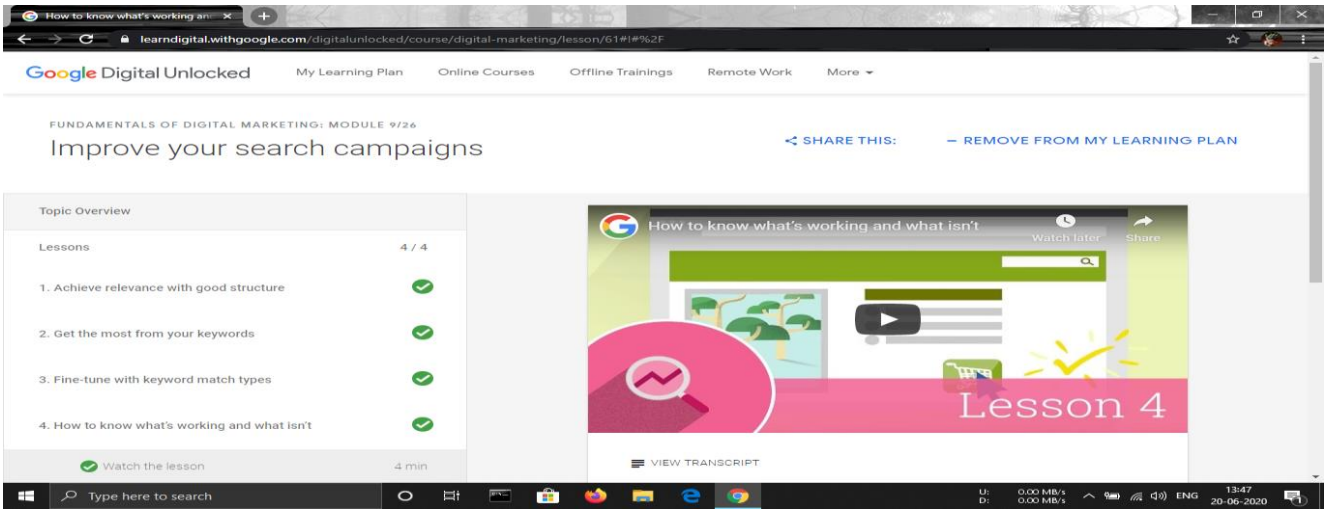
- The **C preprocessor** uses the **# directives** to make substitutions in program source code before compilation.
- For example, the line **#include <stdio.h>** is replaced by the contents of the **stdio.h** header file before a program is compiled.
- **Preprocessor directives** and their uses:
  - ✓ **#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.
- **Formatting Preprocessor Directives** - When using preprocessor directives, the # must be the first character on a line. But there can be any amount of white space before # and between the # and the directive. If a # directive is lengthy, you can use the \ continuation character to extend the definition over more than one line.
- In addition to defining your own macros, there are several standard predefined macros that are always available in a C program without requiring the **#define** directive:
  - ✓ **__DATE__:** The current date as a string in the format Mm dd yyyy.
  - ✓ **__TIME__:** The current time as a string in the format hh:mm:ss.
  - ✓ **__FILE__:** The current filename as a string.
  - ✓ **__LINE__:** The current line number as an int value.
  - ✓ **__STDC__:** 1.
- The **# macro operator** is called the stringification or stringizing operator and tells the preprocessor to convert a parameter to a string constant. White space on either side of the argument are ignored and escape sequences are recognized.
- The **## operator** is also called the token pasting operator because it appends, or "pastes", tokens together.

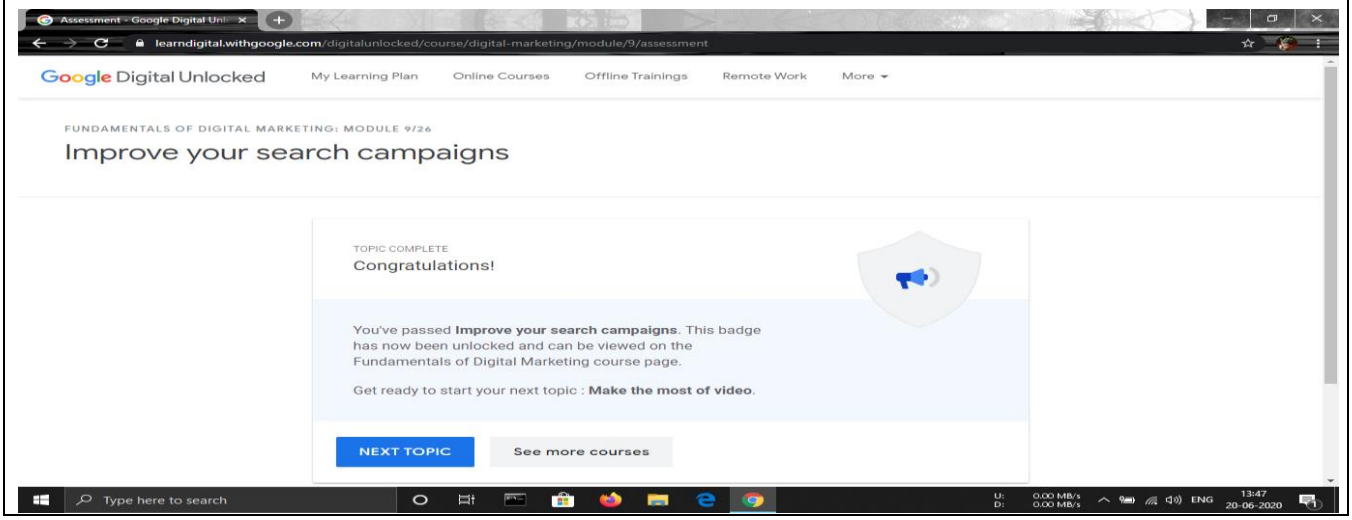
CERTIFICATE - C Tutorial by SOLOLEARN

Date:	20/06/2020	Name:	Abhishek M Shastry K
Course:	Google Digital Unlocked: Fundamentals of digital marketing	USN:	4AL17EC002
Topic:	1] Improve your search campaigns • Achieve relevance with good structure • Get the most from your keywords • Fine-tune with keyword match types • How to know what's working and what isn't	Semester & Section:	6th 'A'
Github Repository:	AbhishekShastry-Courses		

AFTERNOON SESSION DETAILS

Image of session





Report

Achieve relevance with good structure

- When it comes to relevance, search engines also consider the first web page people see after they click on an ad. This is called the landing page. So, what does this mean for you? Well, just as your advert should be relevant to the words a person just searched for, your landing page should be relevant to the advert a person clicks on.
- For starters, search engines reward relevant adverts with higher positions on the search results page. What's more, if your adverts are more relevant than your competitors', you might be able to get the same amount of traffic for a lower price.
- Within the account are campaigns. Each campaign controls important decisions, like the daily budget, the areas or countries where ads can appear, and the advertising networks you want to use.
- Within each campaign, you can create multiple ad groups. These are collections of keywords and the ads that go with them.
- Structuring your account in this organized way helps ensure you show the most relevant ads. Let's go through another example with the photographer to bring this to life. Imagine you're the photographer, and you specialize in various types of photography. Let's say weddings, baby photos and family portraits. Each of these specialties contain different products, so you decide to split them into separate campaigns.

Get the most from your keywords

- By organizing these terms into ad groups, you can write relevant ads for each group of keywords. That takes care of the relevant keywords, but what about the ones that didn't seem so relevant? For example, we can see lots of people are also searching for [pet portraits pencil], which suggests they're looking for something else.
- There are also many who are searching for watercolor pet portraits, and horse portraits, neither of which you offer. In situations like these, you should use "negative keywords" to prevent your ads from appearing when people search for things that aren't relevant to your business. For example, [-pencil], [-pastel] and [-horse].
- These negative keywords will block your ads any time a search contains one or all of them.

Fine-tune with keyword match types

- Most of the time, broad match is useful. It means that you don't have to add every variation of the keyword you'd like to target, like singulars, plurals and misspellings.
- But this flexibility also means that sometimes, search engines show your ads for keywords that aren't relevant to your business. Using keyword match types can help.
- Phrase match tells Google Ads or Bing Ads that adverts can't be displayed unless the search includes the entire phrase. So, if someone searches for "London portrait photographer" that's great-your ads can show up! Minor variations, like plurals, are included. This means that a search for "London portrait photographers" can also trigger your ad.
- Now, if someone searches for portrait photographer, your ad can't appear because it doesn't match the keyword exactly. Along the same lines, a search for London photographer also won't trigger your ad.
- Unlike phrase match, the ad can't display if the searcher includes additional words. But minor variations, like plurals, can still trigger the ad. As you change keywords from broad match, to phrase, to exact, it restricts the opportunities for ads to display. Your best bet is to try to find a match type balance, allowing ads to show to likely prospects, but blocking ads when you think success is unlikely.

How to know what's working and what isn't

- One of the best things about SEM is that you can measure the value you're getting from your campaigns. To do this, you track conversions, the key actions you want website visitors to take using tools that search engines like Google or Bing provide. We'll get to those in just a minute.
- You could also have a link people can click to receive your rates via email. That's another way potential customer can become paying customers, so you should track that as a conversion as well. In these two examples, we've mentioned a handful of different conversions: successful transactions, contact form submissions, and downloads. And there are plenty of other possibilities. So how can you actually track these conversions? Well, you can use tools provided by search engines. These allow you, or whoever is managing your website for you, to place a small piece of code on certain pages of your website.