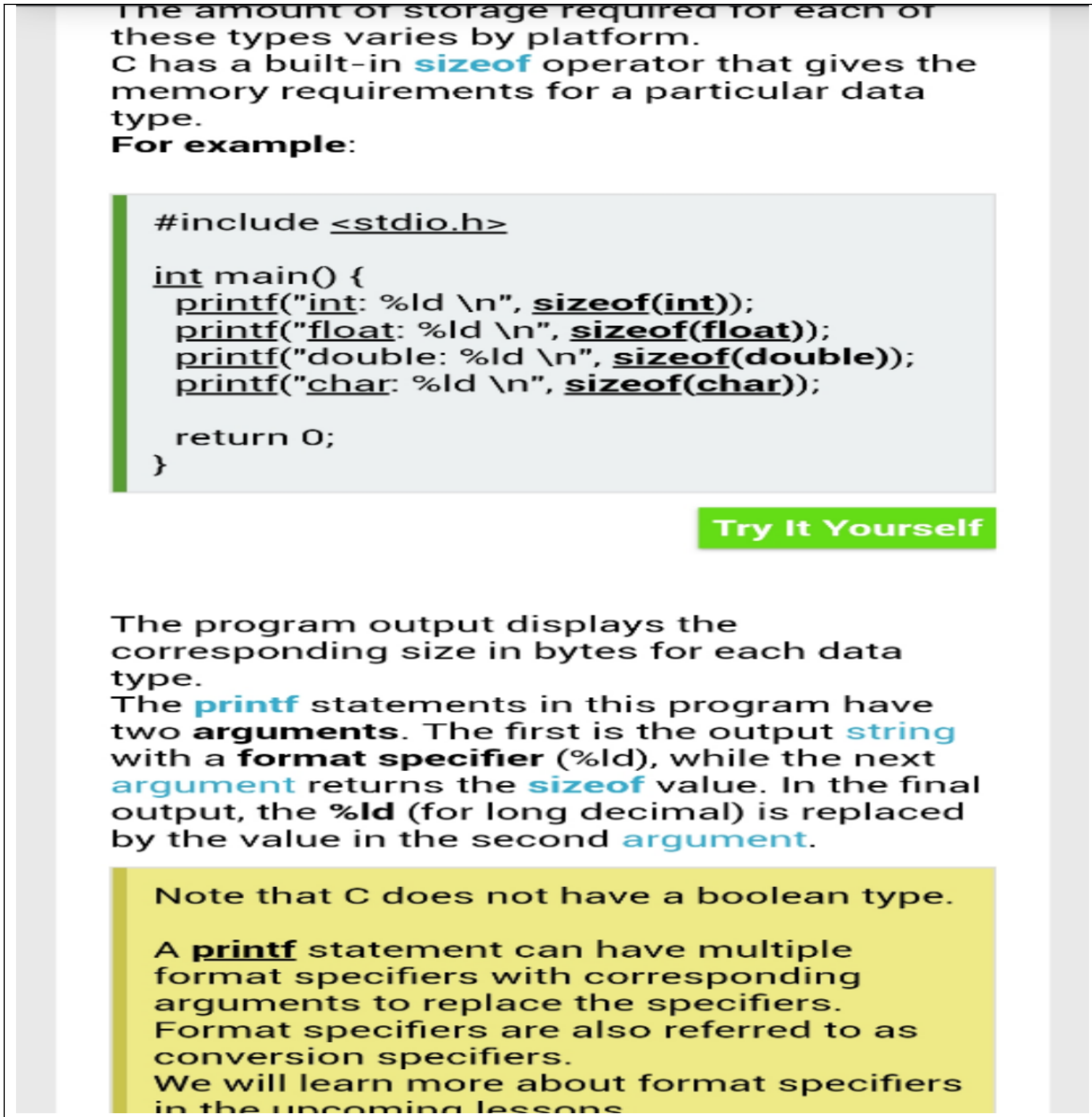
DAILY ASSESSMENT FORMAT

Date:	20-06-2020	Name:	Jagadeesha Hegde
Course:	C programming	USN:	4AL17EC036
Topic:	Files and Error Handling,Pre-Processor	Semester & Section:	6th A-sec
Github Repository:	Jagadeesha-036		

FORENOON SESSION DETAILS
Image of session



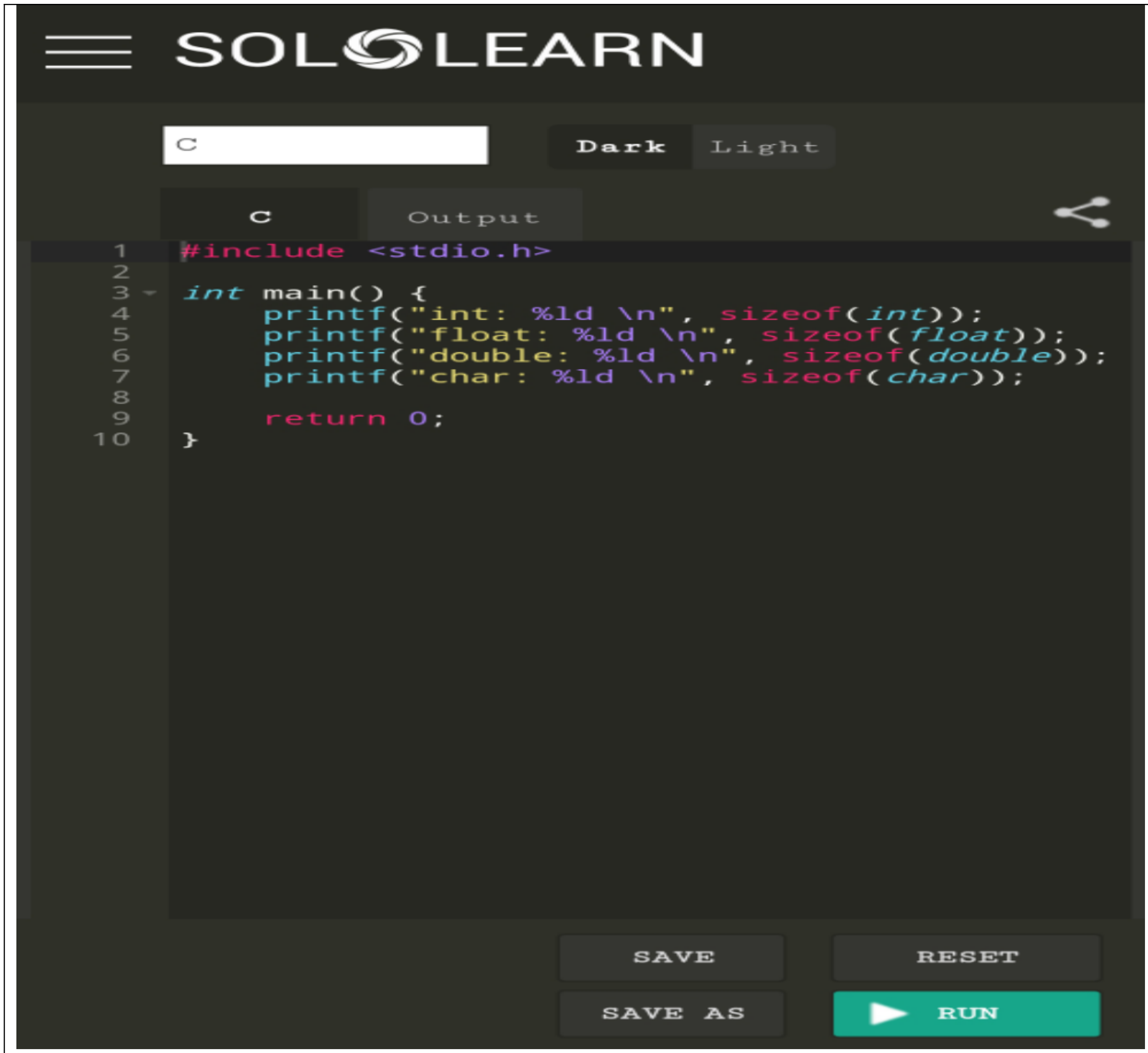
The amount of storage required for each of these types varies by platform.
C has a built-in sizeof operator that gives the memory requirements for a particular data type.
For example:

```
#include <stdio.h>

int main() {
  printf("int: %ld \n", sizeof(int));
  printf("float: %ld \n", sizeof(float));
  printf("double: %ld \n", sizeof(double));
  printf("char: %ld \n", sizeof(char));

  return 0;
}
```

The program output displays the corresponding size in bytes for each data type.
The printf statements in this program have two **arguments**. The first is the output string with a **format specifier** (%ld), while the next argument returns the sizeof value. In the final output, the **%ld** (for long decimal) is replaced by the value in the second argument.

Note that C does not have a boolean type.

A **printf** statement can have multiple format specifiers with corresponding arguments to replace the specifiers.
Format specifiers are also referred to as conversion specifiers.
We will learn more about format specifiers in the upcoming lessons



```c
#include <stdio.h>

int main() {
     printf("int: %ld \n", sizeof(int));
     printf("float: %ld \n", sizeof(float));
     printf("double: %ld \n", sizeof(double));
     printf("char: %ld \n", sizeof(char));

     return 0;
}
```

Report – Report can be typed or hand written for up to two pages.

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.