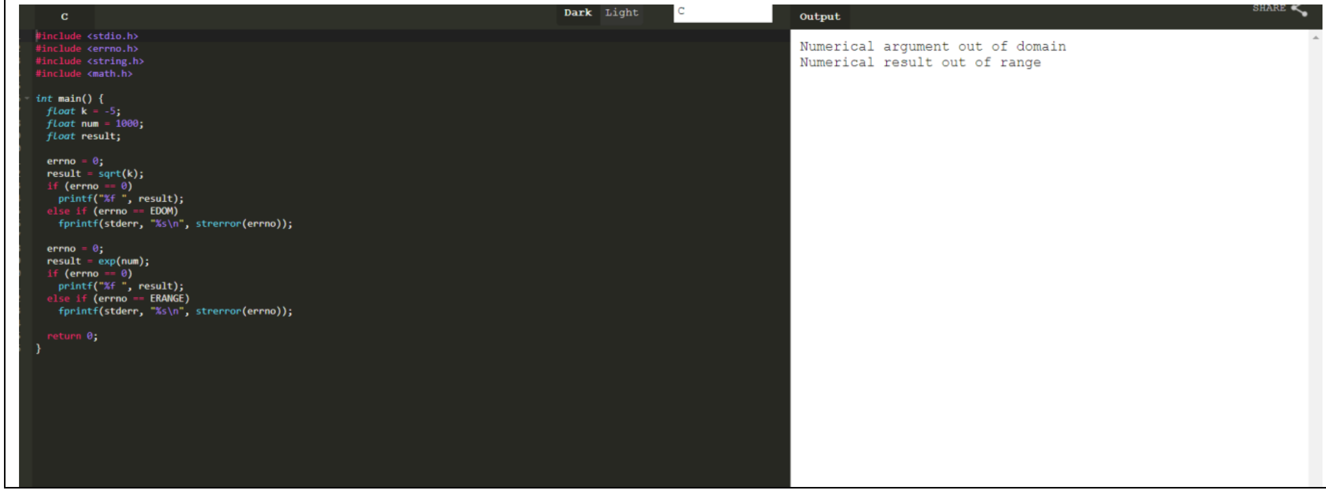
DAILY ASSESSMENT FORMAT

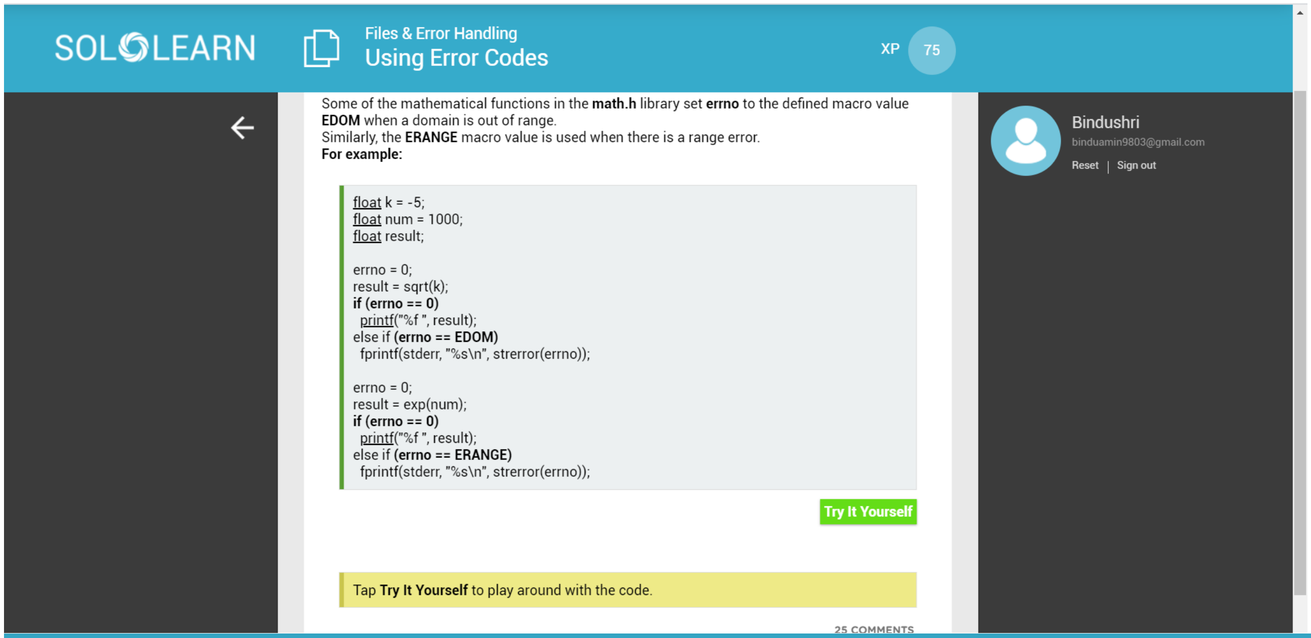
Date:	**20-06-2020**	**Name:**	**BINDUSHRI**
Course:	**C programming**	**USN:**	**4AL17EC011**
Topic:	**Files and error handling** **Error handling**		**6th A**
Github Repository:	**Bindushri**		

FORENOON SESSION DETAILS



```c
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <math.h>

int main() {
  float k = -5;
  float num = 1000;
  float result;

  errno = 0;
  result = sqrt(k);
  if (errno == 0)
    printf("%f ", result);
  else if (errno == EDOM)
    fprintf(stderr, "%s\n", strerror(errno));

  errno = 0;
  result = exp(num);
  if (errno == 0)
    printf("%f ", result);
  else if (errno == ERANGE)
    fprintf(stderr, "%s\n", strerror(errno));

  return 0;
}
```

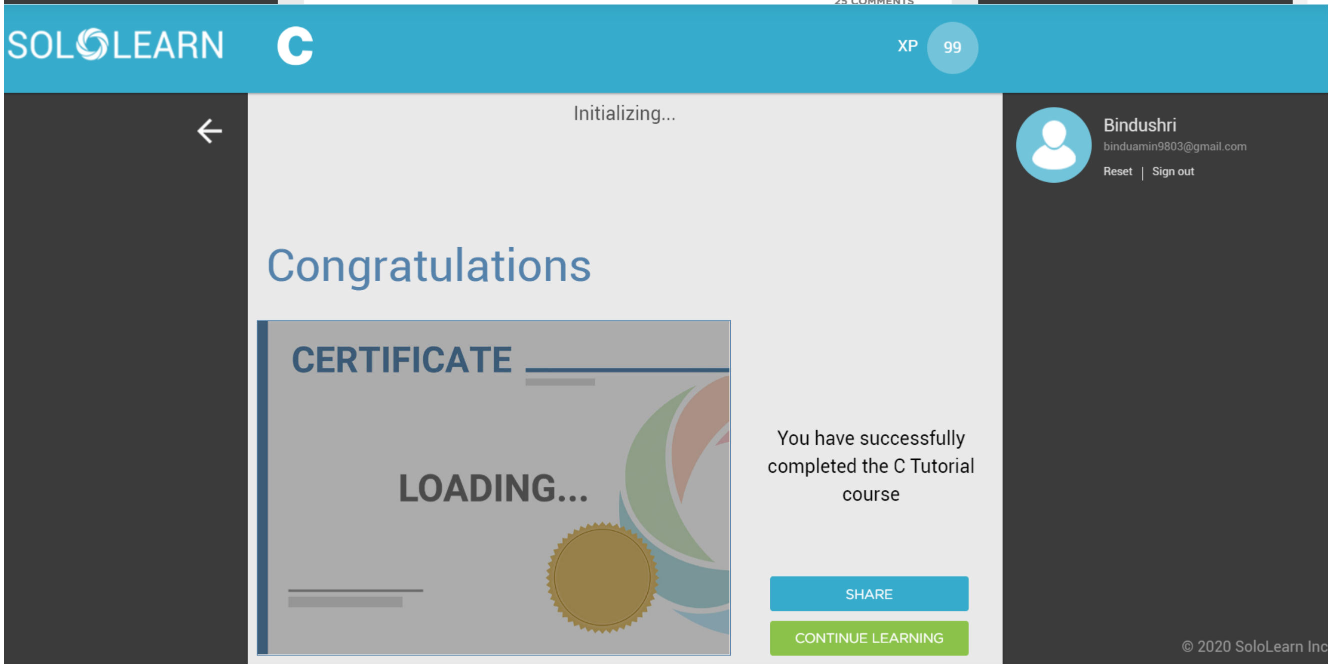
```
Numerical argument out of domain
Numerical result out of range
```



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.
For example:

```
float k = -5;
float num = 1000;
float result;

errno = 0;
result = sqrt(k);
if (errno == 0)
  printf("%f ", result);
else if (errno == EDOM)
  fprintf(stderr, "%s\n", strerror(errno));

errno = 0;
result = exp(num);
if (errno == 0)
  printf("%f ", result);
else if (errno == ERANGE)
  fprintf(stderr, "%s\n", strerror(errno));
```

Tap **Try It Yourself** to play around with the code.



Congratulations

You have successfully completed the C Tutorial course

```
int cap=1;
```

20/06/2020

Files & Error Handling

Working with files

Accessing files

An external file can be opened, read from, and written to in C program.

→ stdio.h library includes filehandling function

→ FILE typedef for defining a file pointer

r - open for reading
w - open for writing
a - open for append
r+ → open for reading & writing

w+ → overwriting
a+ → appending reading & writing.

```
#include <stdio.h>
int main() {
FILE *fptr;
fptr = fopen ("myfile.txt", "w");
if (fptr==NULL){
printf ("Error opening file");
return-1;
}
fclose (fptr);
return0;
```

Reading from a file.

the stdio.h library also includes functions for reading from an open file.

fgetc (fp) → Returns next character from file pointed by fp

fgets(buff,n,fp) reads n-1 characters

fscanf(fp, conversion_specifiers, vars) → Reads characters from ~~char~~ the file by fp & assigns input to a list of variable pointers vars using conversion_specifiers.

writing to a file

the stdio.h library also includes functions for writing to a file.
'\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 to the file pointed to by fp.

Binary file I/O

rb - open for reading
wb - open for writing.
ab - open for append.
* rb+ open for reading & writing beginning
* wb+ open for reading & writing overwriting
* ab+ open for reading & writing appending to file

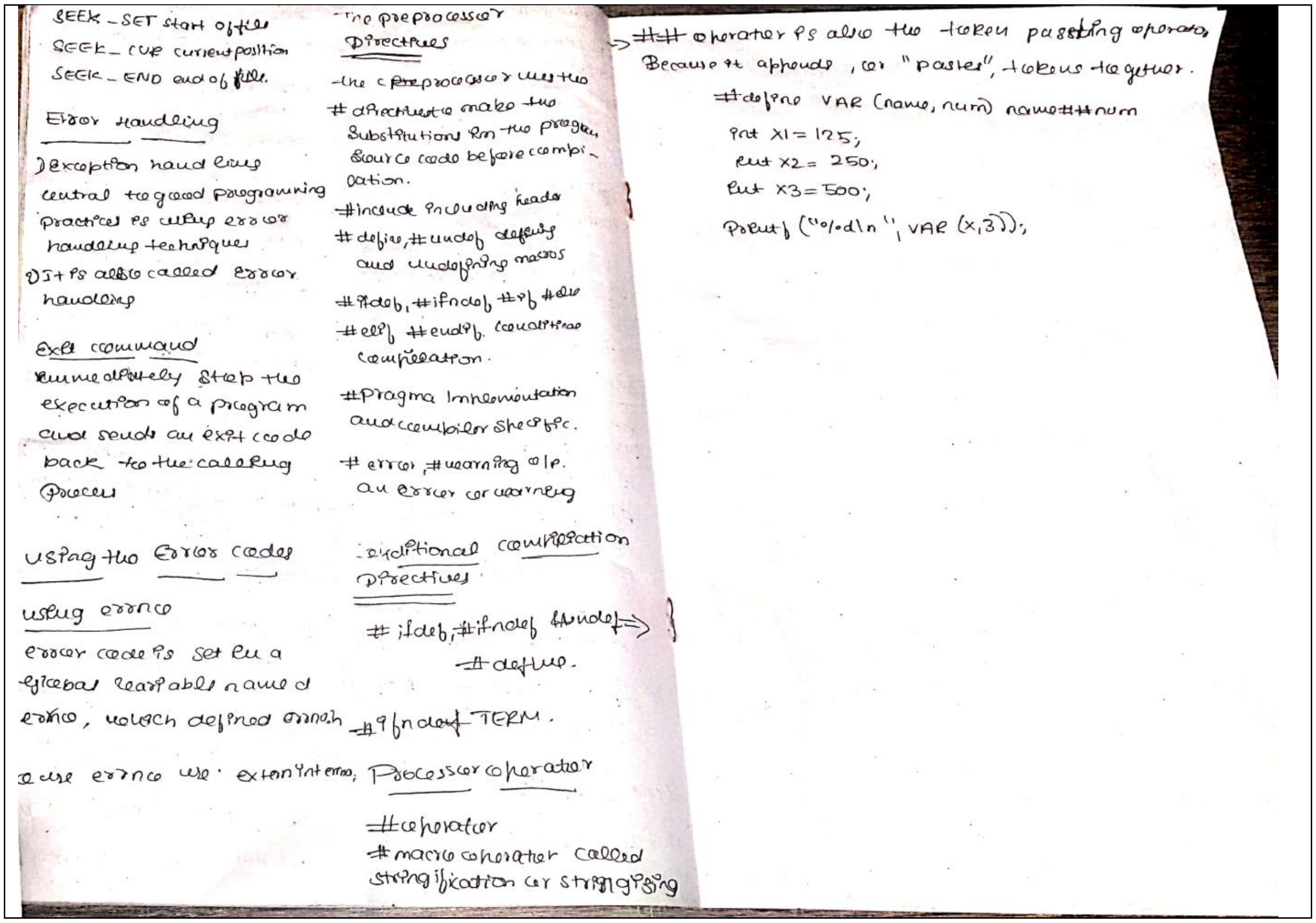
Binary File I/O

```
FILE *fptr
int arr[10];
int x[10];
int k;
for (k=0;k<10; k++)
arr[k]=k;
fptr = fopen ("datafile.bin","w
fwrite(arr,sizeof (arr[0]), sizeof(arr)/
                 sizeof (arr [0]), fptr);
fclose (fptr);

fptr= fopen (datafile.bin", "rb")
fread (x,sizeof(arr[0]),sizeof(a
          sizeof(arr[0]),fptr);
fclose(fptr);

for (k=0; k<10; k++)
printf ("%d", x[k]);
```

* controlling the file pointer

ftell(fp) Returns long int value corresponding to the fp file pointer position.

fseek(fp,num_bytes, from_po



SEEK_SET start of file
SEEK_CUR current position
SEEK_END end of file.

Error Handling

1) Exception handling central to good programming practices is using error handling techniques
2) It is also called error handling

Exit command

Immediately stop the execution of a program and sends an exit code back to the calling Process

Using the Error codes

using errno

error code is set in a global variable named errno, which defined errno.h

To use errno we: extern int errno;

- The preprocessor Directives

- The C ~~pre~~preprocessor uses the # directive to make the substitutions in the program source code before compilation.

#include including header

#define, #undef defining and undefining macros

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

#pragma Implementation and compiler specific.

#error, #warning o/p. an error or warning

Conditional compilation Directives

ifdef, #ifndef #undef => }
#define.

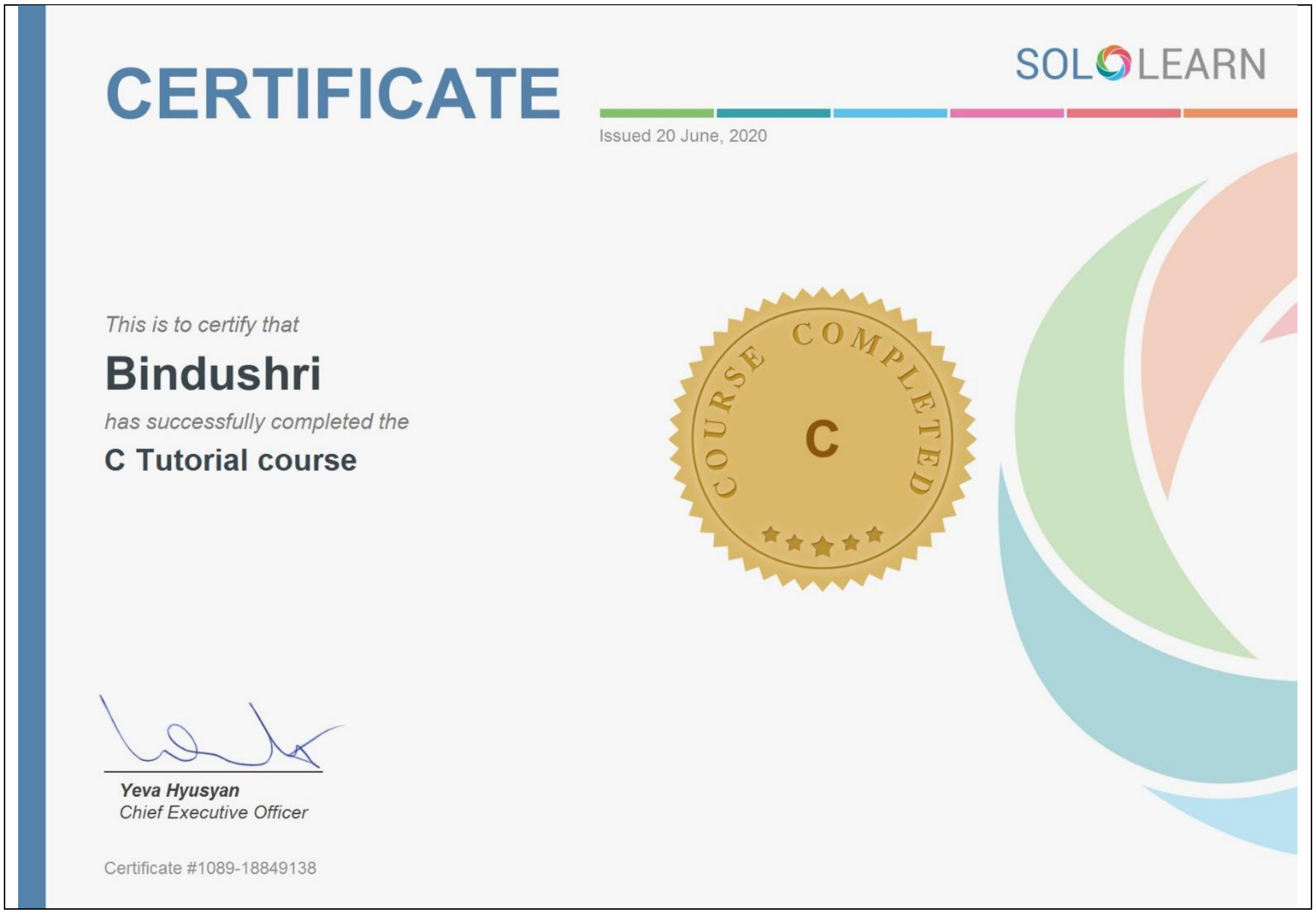
#ifndef TERM.

Processor operator

#operator
macro operator called stringification or stringizing

-> ## operator is also the token pasting operator, Because it appends, or "pastes", tokens together.

```
#define VAR (name, num) name##num
int x1 = 125;
int x2 = 250;
int x3 = 500;
printf ("%d\n", VAR (x,3));
```



CERTIFICATE

SOLOLEARN

Issued 20 June, 2020

This is to certify that

Bindushri

has successfully completed the

C Tutorial course

COURSE COMPLETED

C

[illegible]

Yeva Hyusyan
Chief Executive Officer

Certificate #1089-18849138