Embedded Systems

Task 8

by Mohammed Elahmady

Task 8 Objects

- Task 8 Objects
  - What is a macro in C, and how is it defined?
  - What is the difference between macros and functions?
  - What do #ifdef, #ifndef, and #endif do?
  - What does malloc() do, and what type does it return?
  - What is the difference between malloc, calloc, and realloc?
  - Why must we always call free() after dynamic allocation?
  - What is a header guard, and what problem does it solve?

What is a macro in C, and how is it defined?

What is the difference between macros and functions?

What do #ifdef, #ifndef, and #endif do?

What does malloc() do, and what type does it return?

What is the difference between malloc, calloc, and realloc?

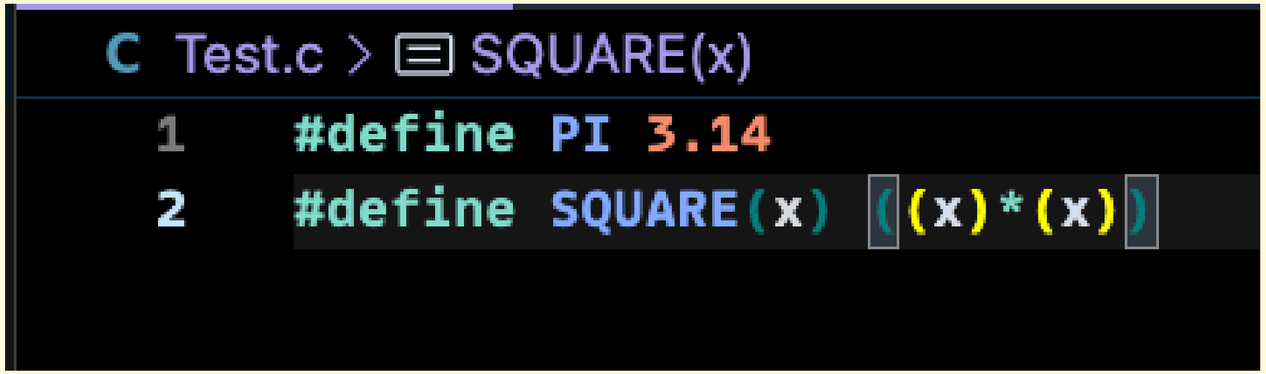
Why must we always call free() after dynamic allocation?

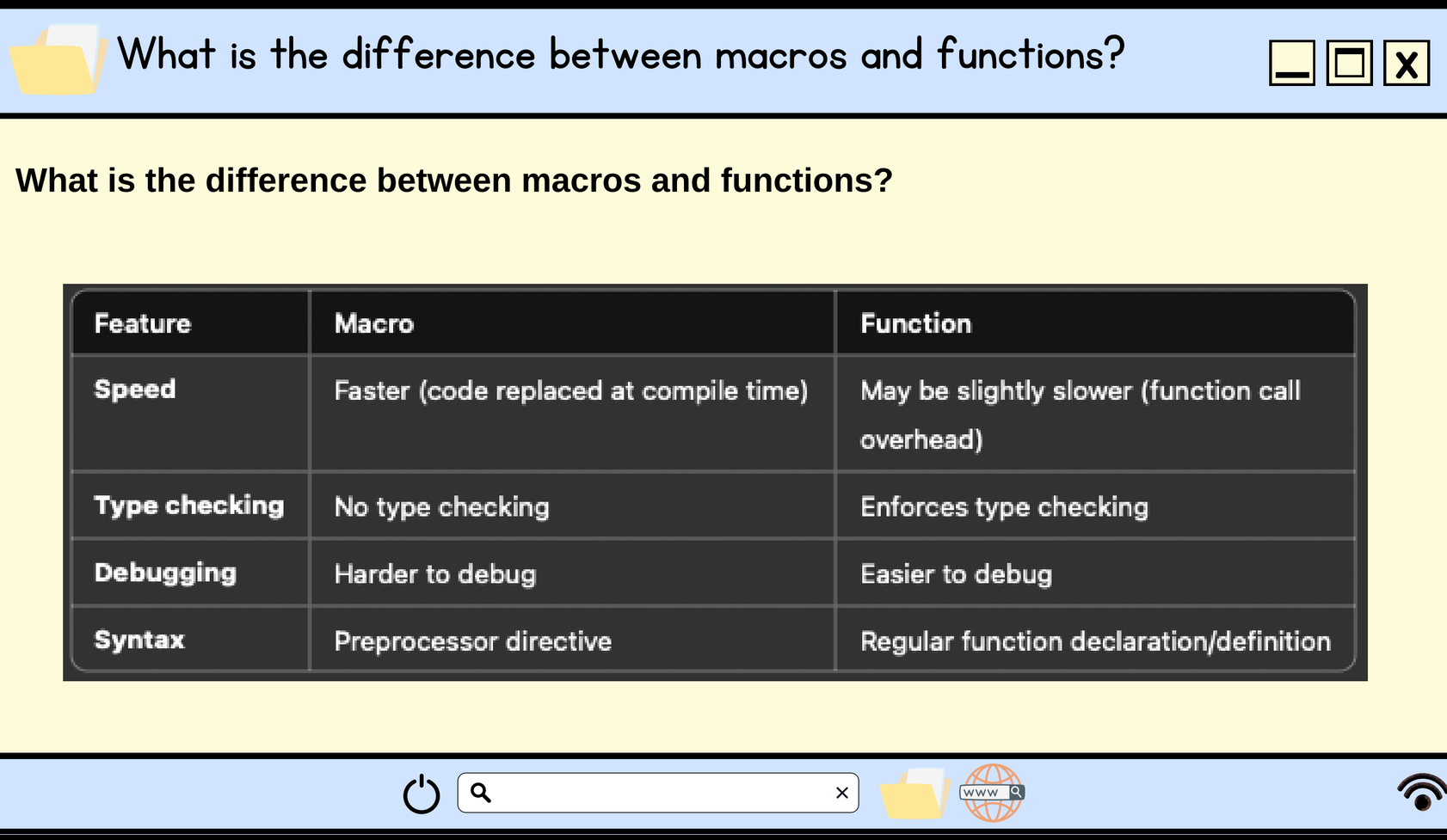
What is a header guard, and what problem does it solve?

What is a macro in C, and how is it defined?

What is a macro in C, and how is it defined?

- A macro in C is a fragment of code which is given a name. Whenever that name is used, the compiler replaces it with the actual code defined. It is defined using #define.




What is the difference between macros and functions?
What is the difference between macros and functions?
Feature
Macro
Function
Speed
Faster (code replaced at compile time)
May be slightly slower (function call overhead)
Type checking
No type checking
Enforces type checking
Debugging
Harder to debug
Easier to debug
Syntax
Preprocessor directive
Regular function declaration/definition
www

What do #ifdef, #ifndef, and #endif do?

These are preprocessor directives used to conditionally compile code:

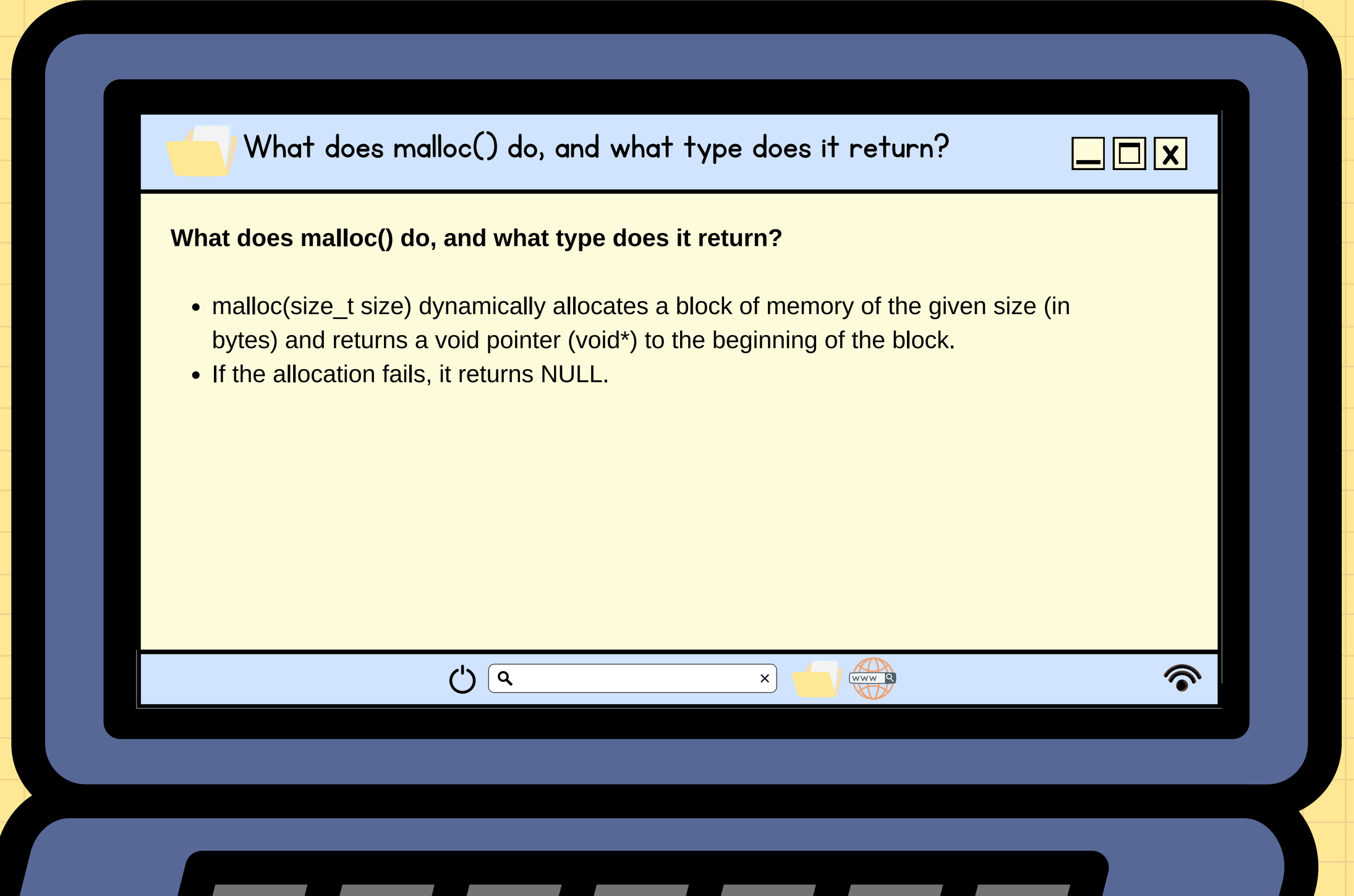
- #ifdef MACRO: Compiles the code only if MACRO is defined.
- #ifndef MACRO: Compiles the code only if MACRO is not defined.
- #endif: Marks the end of the conditional block.

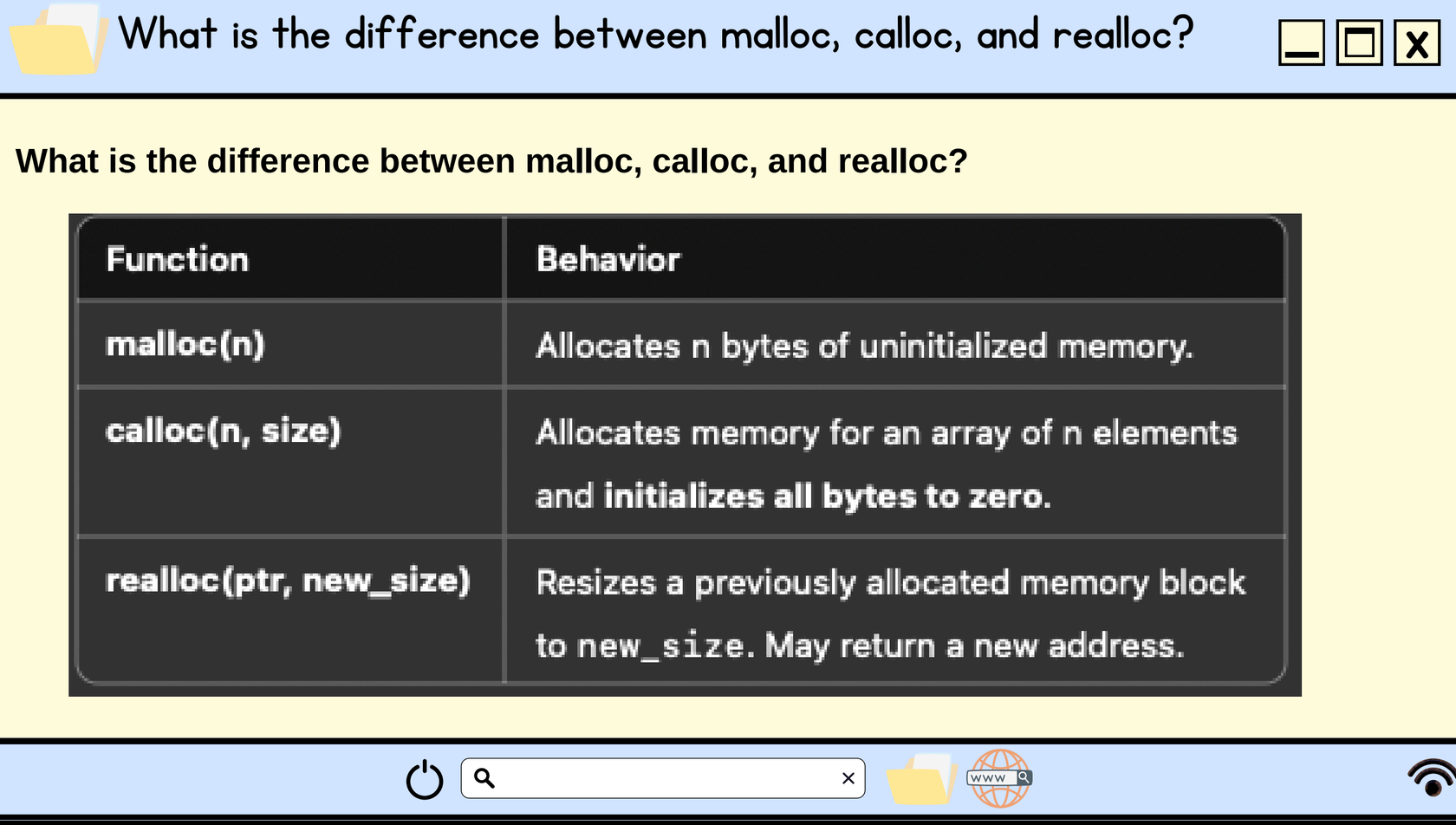
C Test.c > ...

```c
#ifndef MY_HEADER_H
#define MY_HEADER_H

// code

#endif
```


What does malloc() do, and what type does it return?
What does malloc() do, and what type does it return?
malloc(size_t size) dynamically allocates a block of memory of the given size (in bytes) and returns a void pointer (void*) to the beginning of the block.
If the allocation fails, it returns NULL.
www

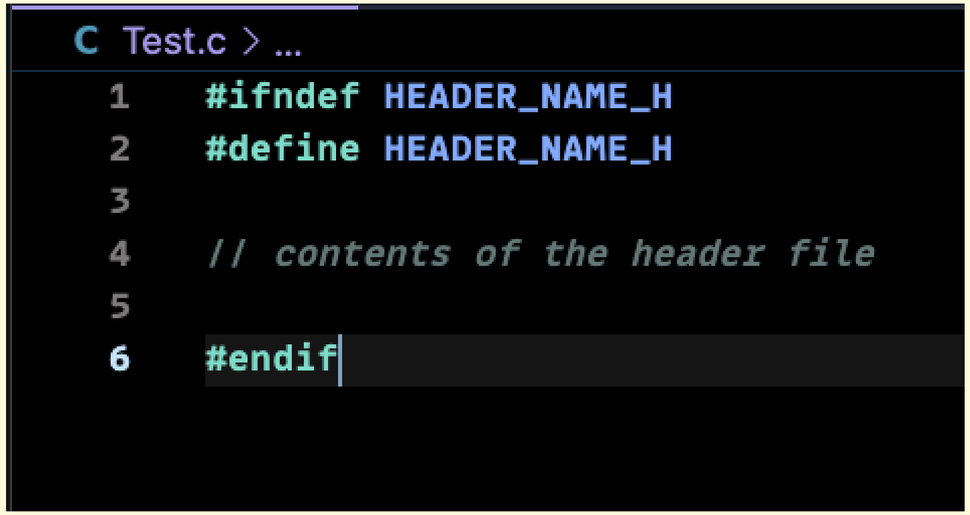

What is the difference between malloc, calloc, and realloc?
What is the difference between malloc, calloc, and realloc?
Function
Behavior
malloc(n)
Allocates n bytes of uninitialized memory.
calloc(n, size)
Allocates memory for an array of n elements and initializes all bytes to zero.
realloc(ptr, new_size)
Resizes a previously allocated memory block to new_size. May return a new address.
www

Why must we always call free() after dynamic allocation?

- To avoid memory leaks. Memory allocated using malloc, calloc, or realloc is not automatically freed when a function exits. If you don’t call free(), it stays allocated until the program ends — which is inefficient and dangerous for long-running applications.

What is a header guard, and what problem does it solve?

- A header guard prevents a header file from being included more than once during compilation. Without it, multiple inclusions can cause redefinition errors.



THANK YOU

Head : Tasnem Sabry

Vice : Ahmed Yasser