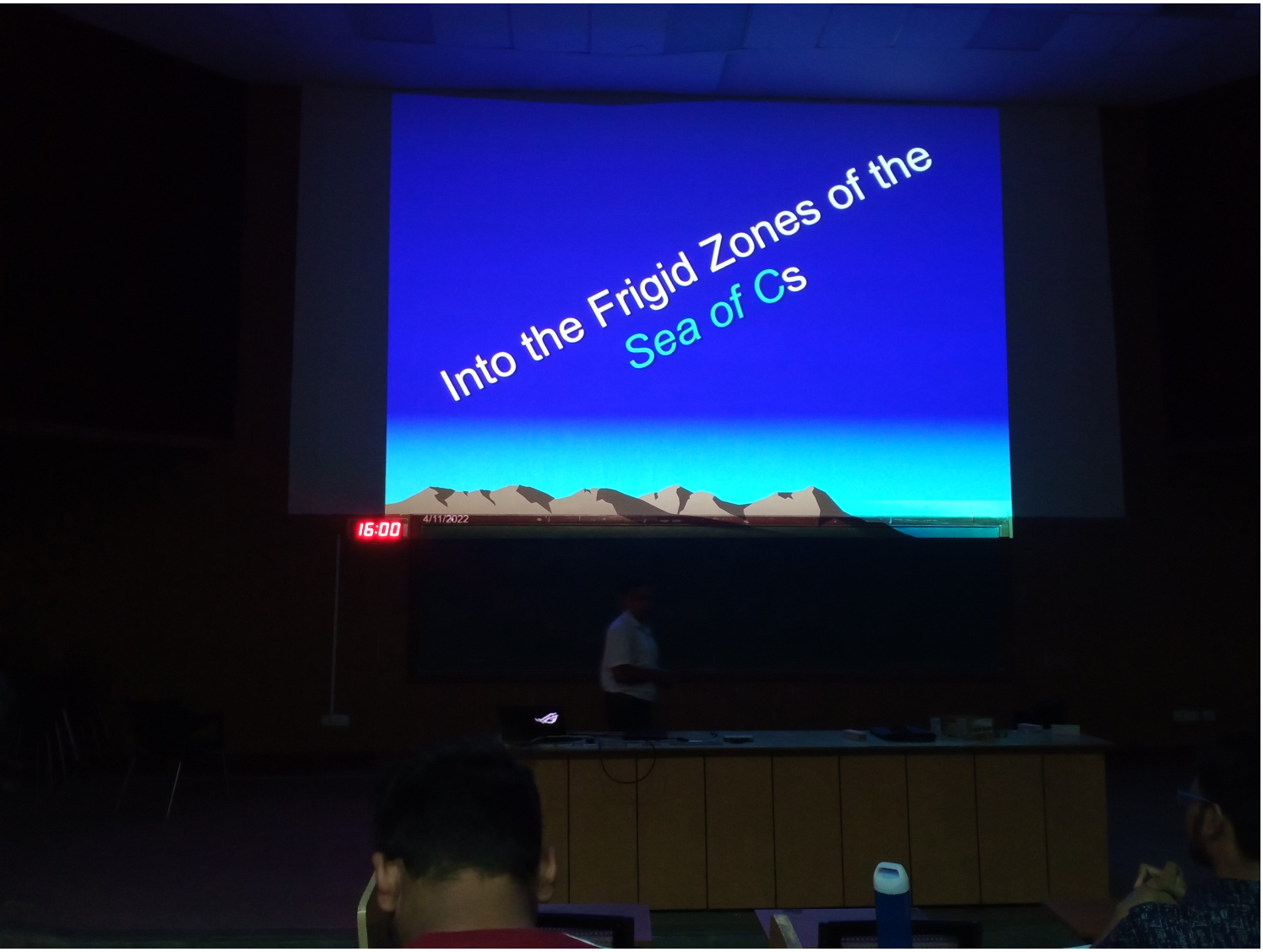
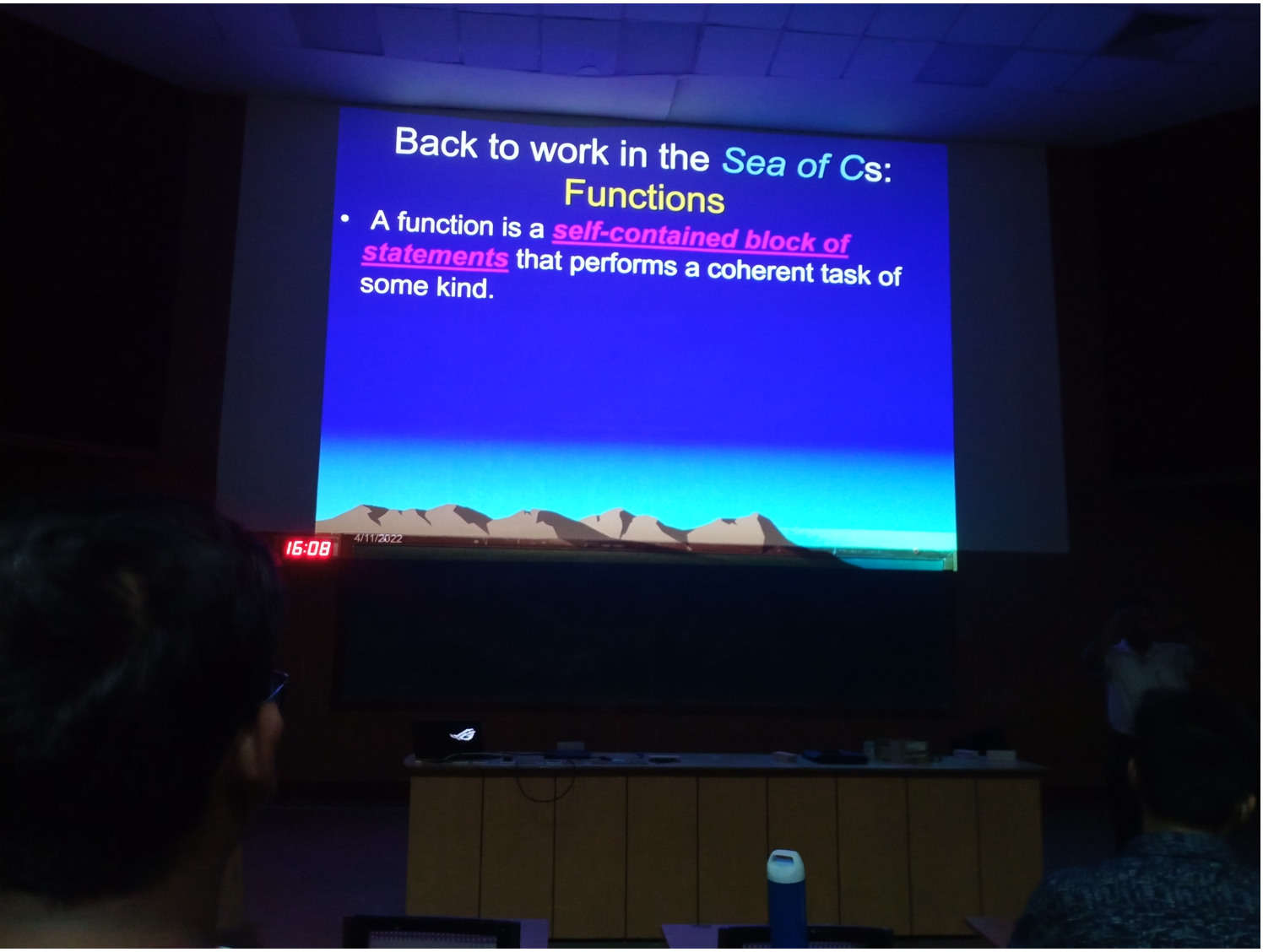
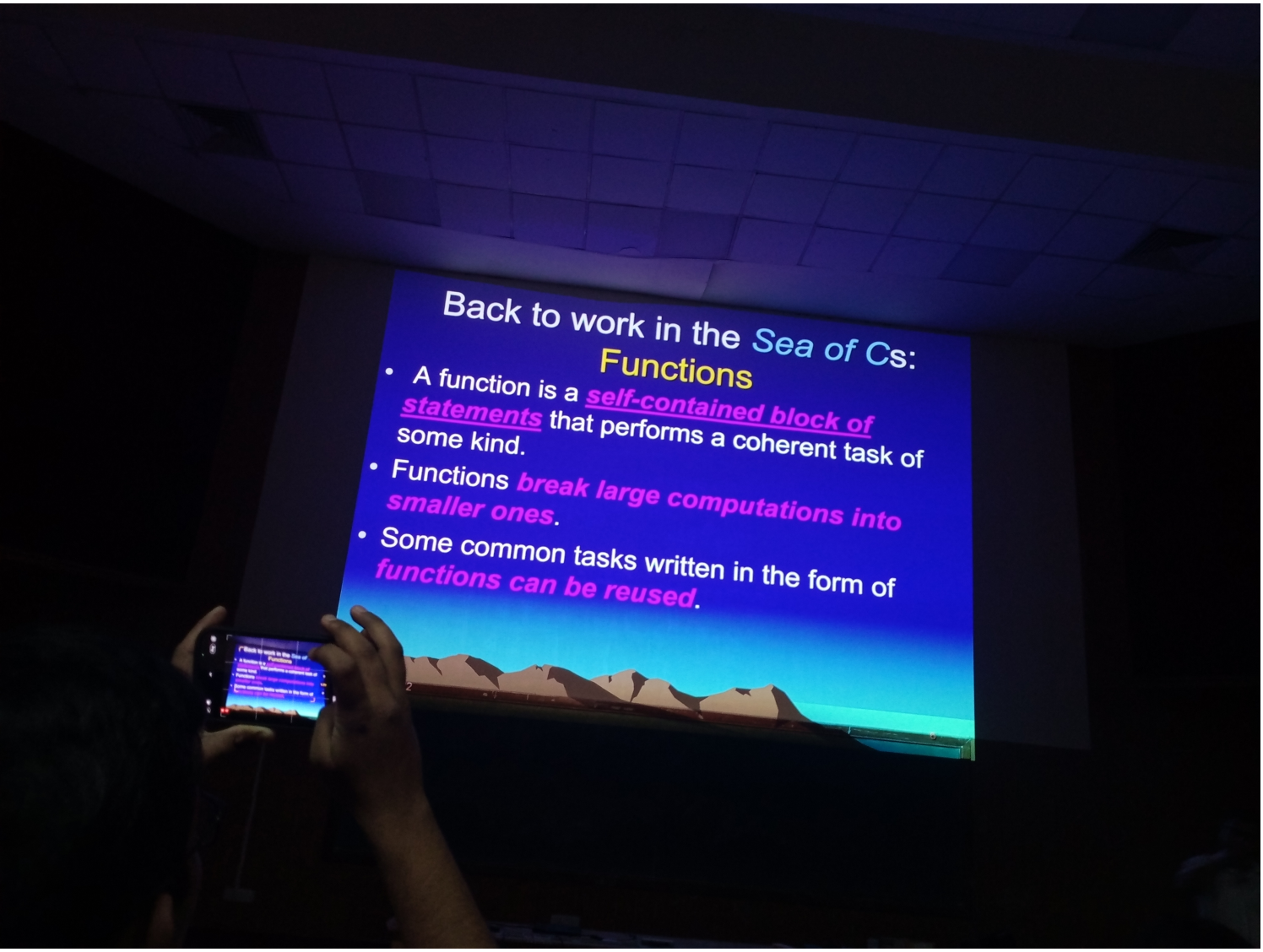
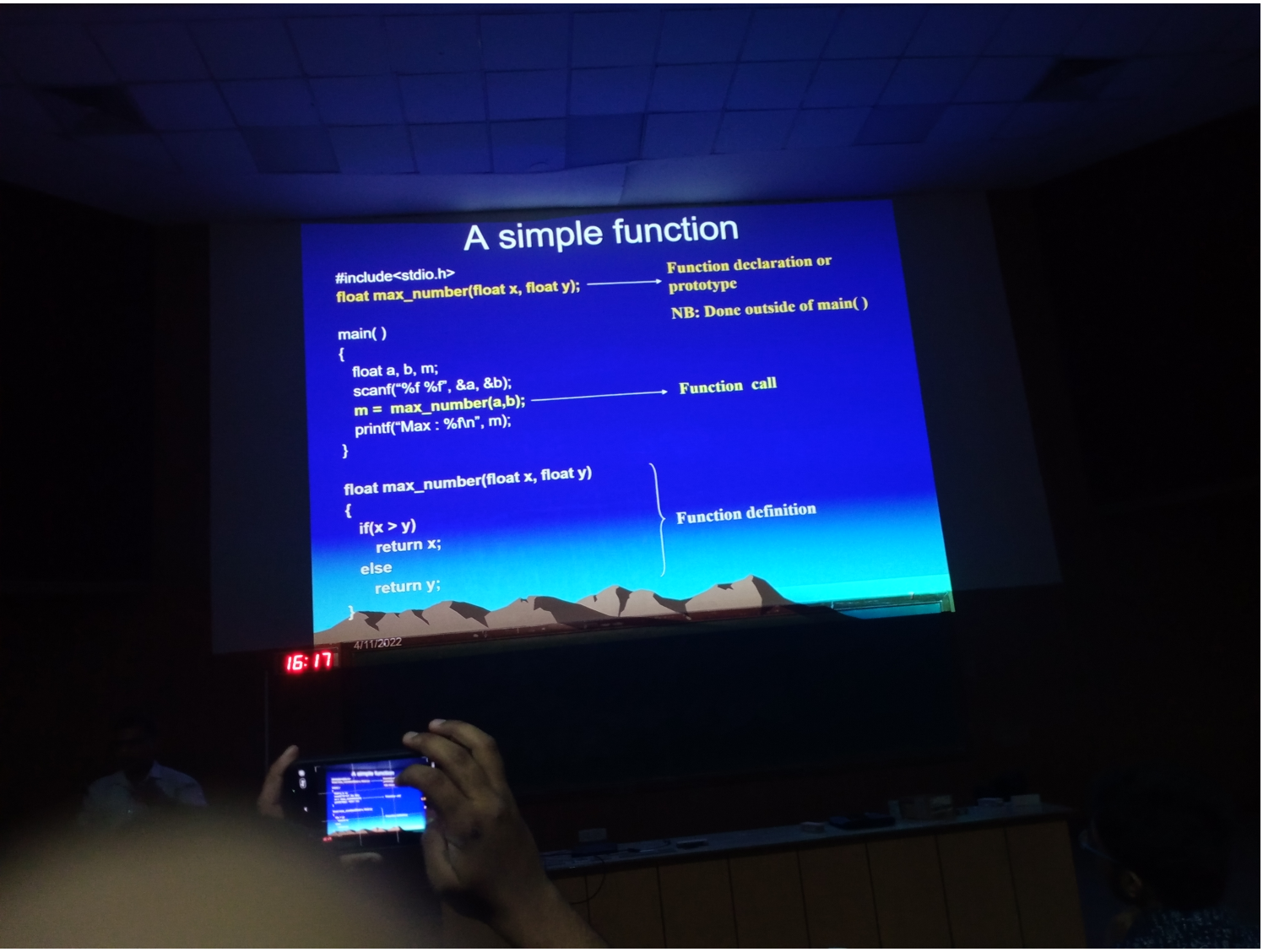
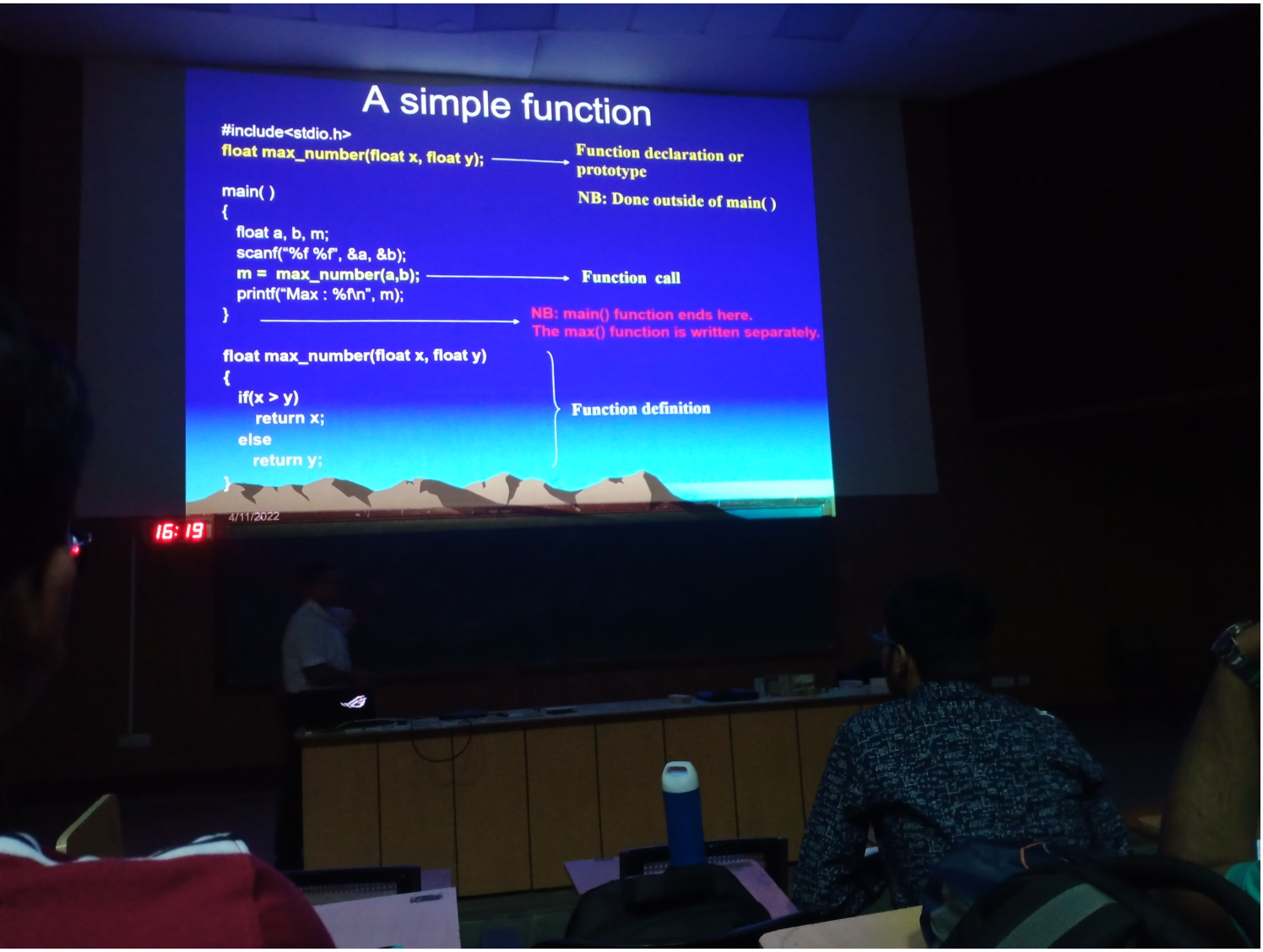
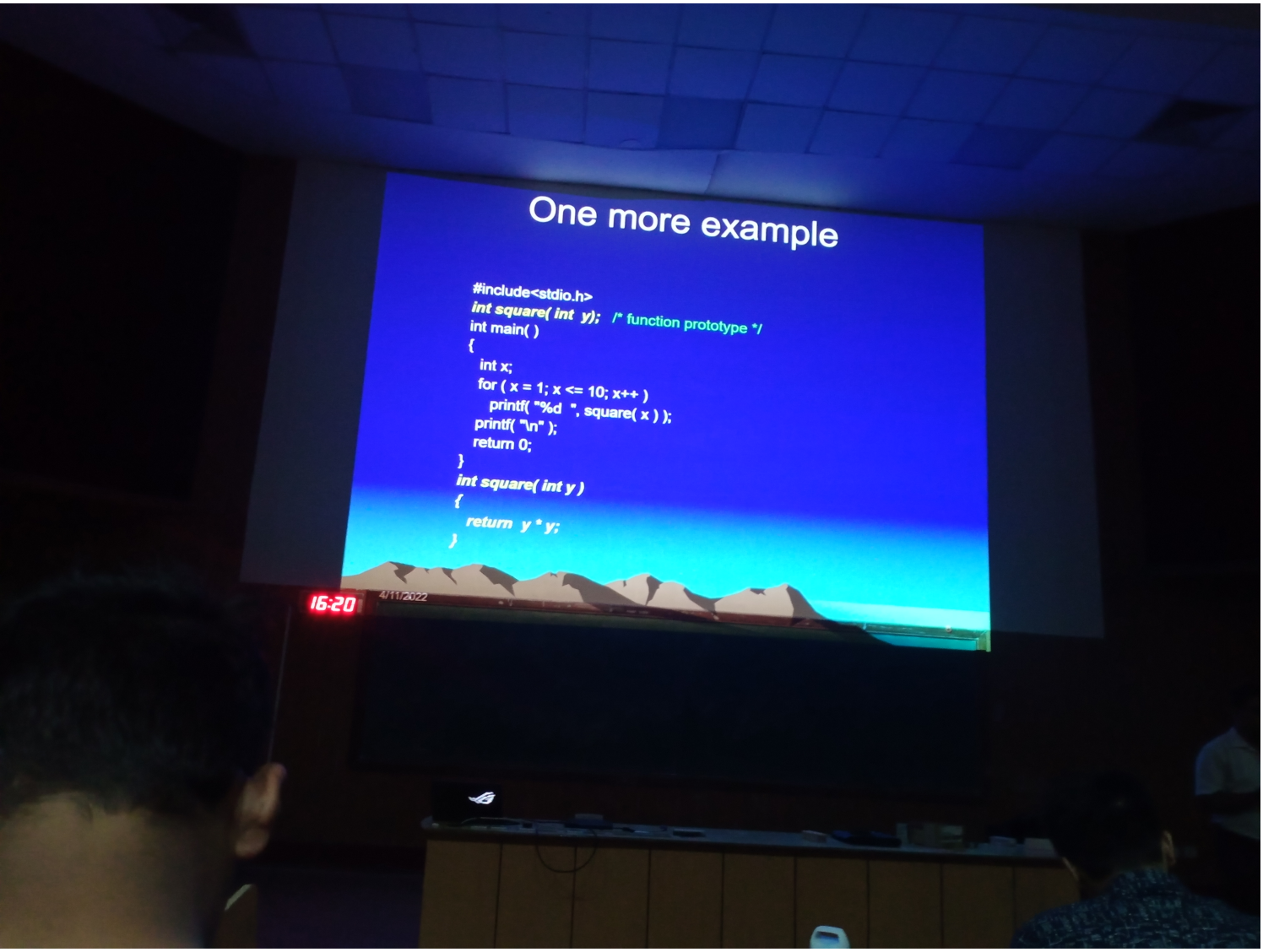

Into the Frigid Zones of the
Sea of Cs
16:00
4/11/2022

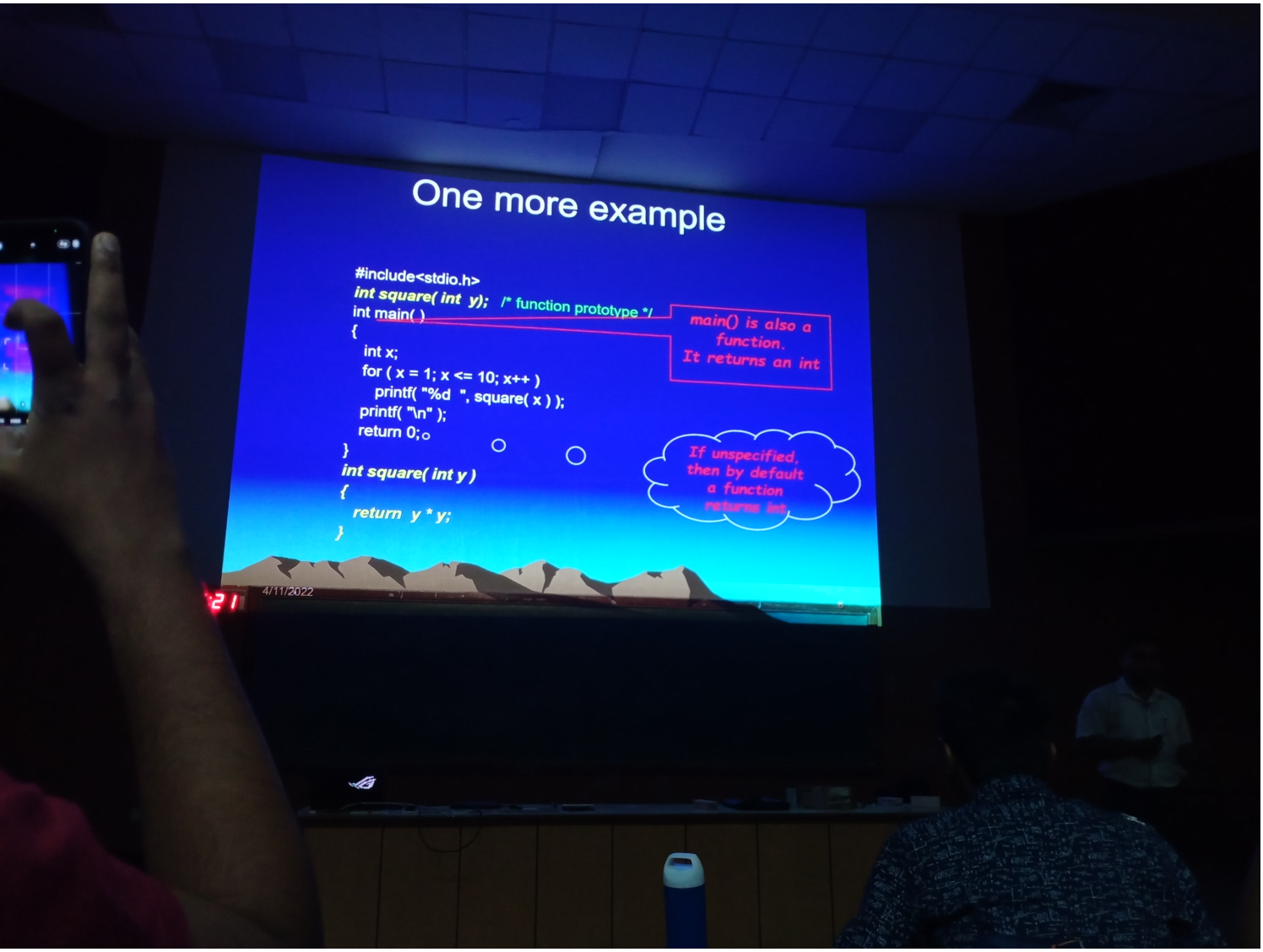

Back to work in the Sea of Cs:
Functions
A function is a self-contained block of statements that performs a coherent task of some kind.
16:08
4/11/2022

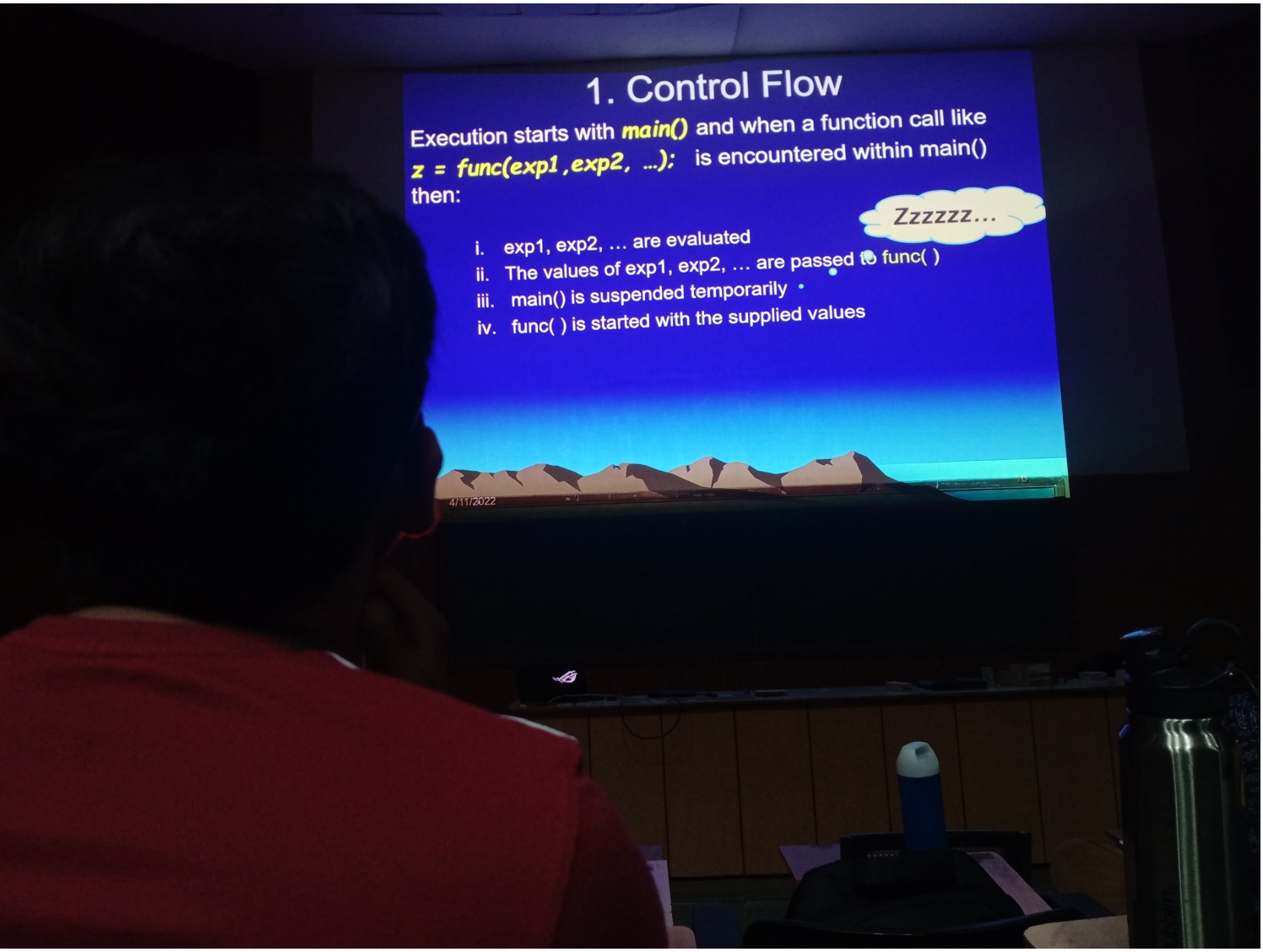

Back to work in the Sea of Cs:
Functions
A function is a self-contained block of statements that performs a coherent task of some kind.
Functions break large computations into smaller ones.
Some common tasks written in the form of functions can be reused.

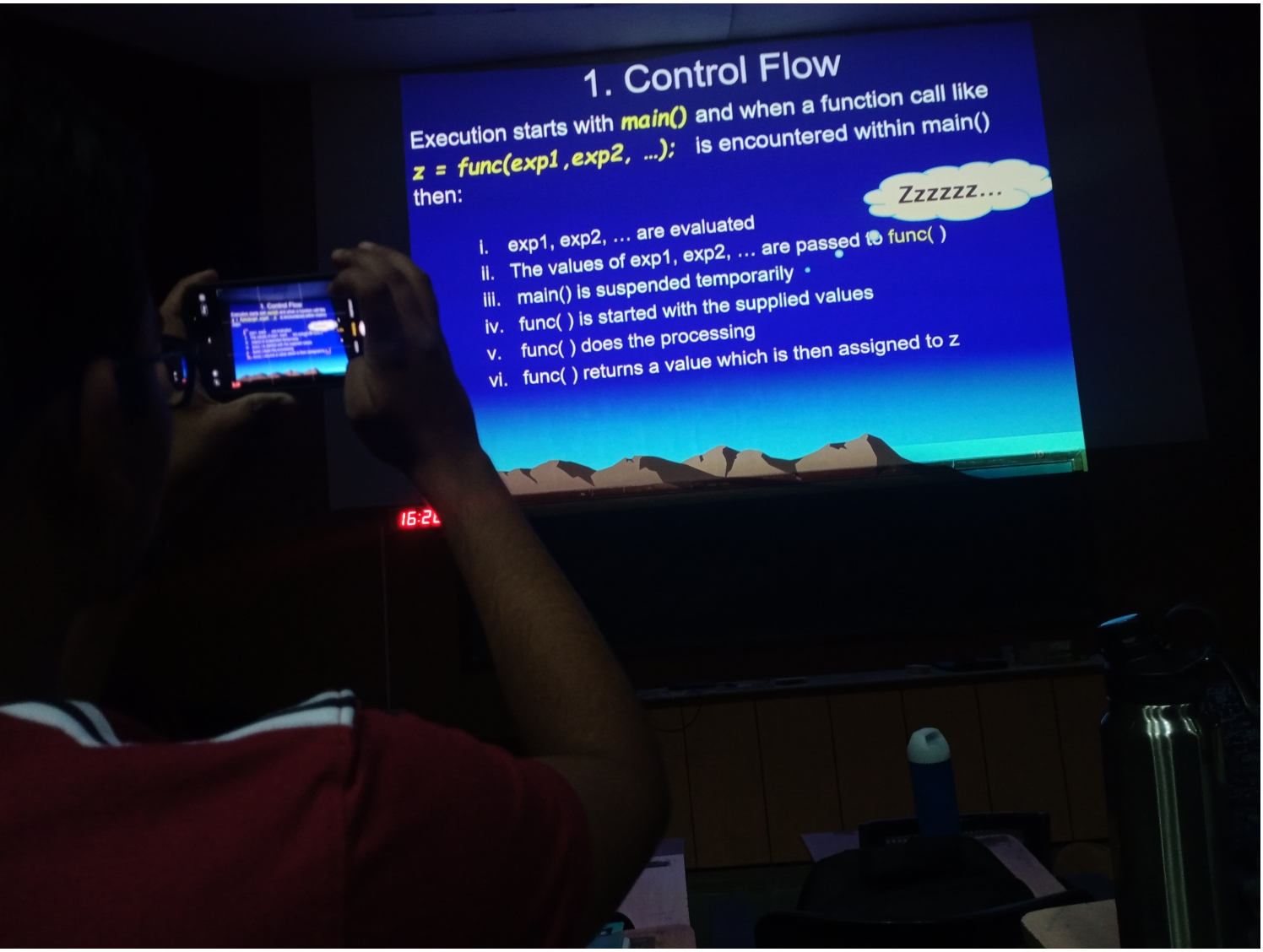

A simple function
#include<stdio.h>
float max_number(float x, float y);
Function declaration or prototype
NB: Done outside of main( )
main( )
{
float a, b, m;
scanf("%f %f", &a, &b);
m = max_number(a,b);
Function call
printf("Max : %f\n", m);
}
float max_number(float x, float y)
{
if(x > y)
return x;
else
return y;
}
Function definition
4/11/2022
16:17

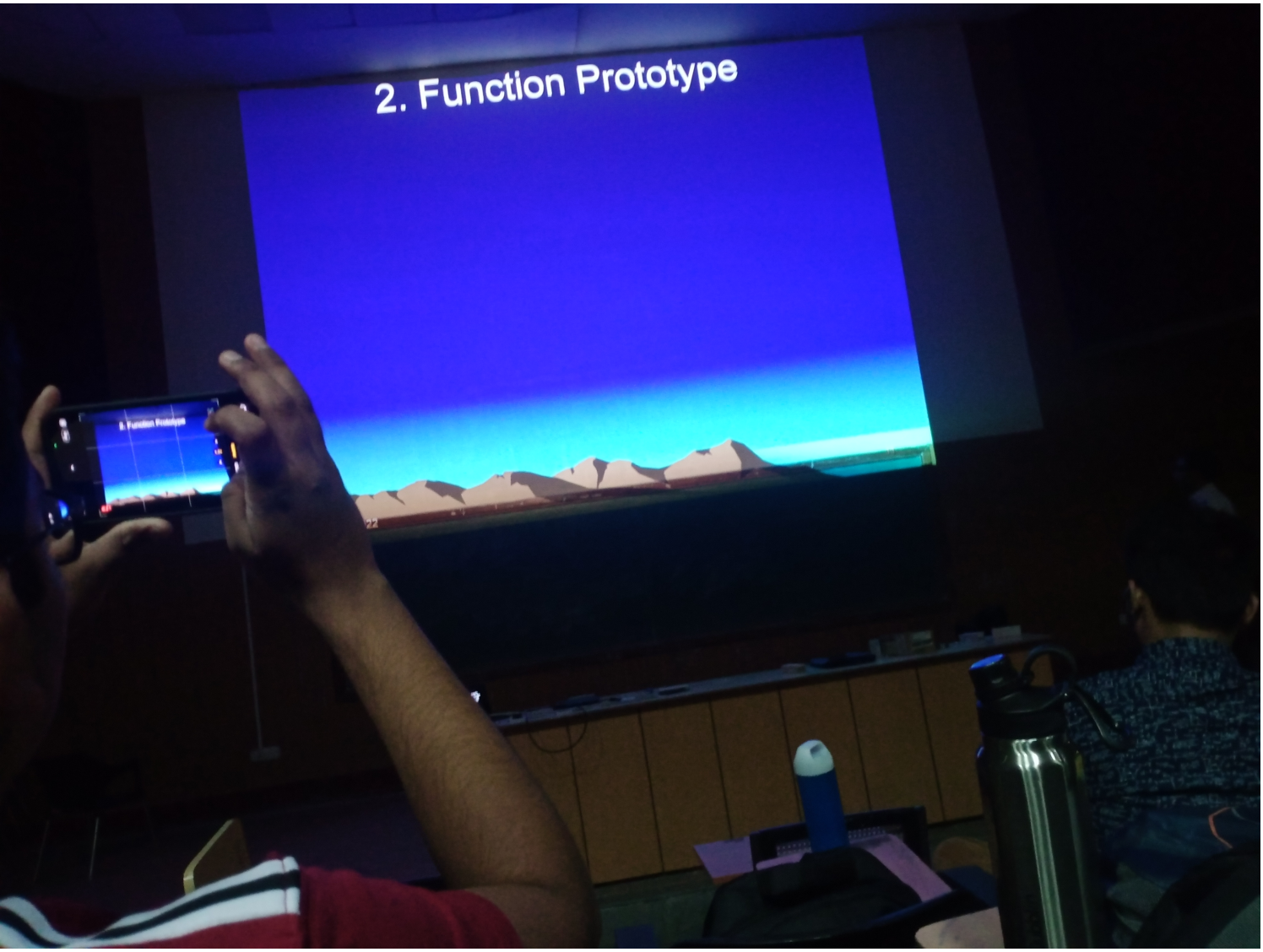

A simple function
#include<stdio.h>
float max_number(float x, float y);
Function declaration or prototype
NB: Done outside of main( )
main( )
{
float a, b, m;
scanf("%f %f", &a, &b);
m = max_number(a,b);
Function call
printf("Max : %f\n", m);
}
NB: main() function ends here.
The max() function is written separately.
float max_number(float x, float y)
{
if(x > y)
return x;
else
return y;
}
Function definition
4/11/2022
16: 19

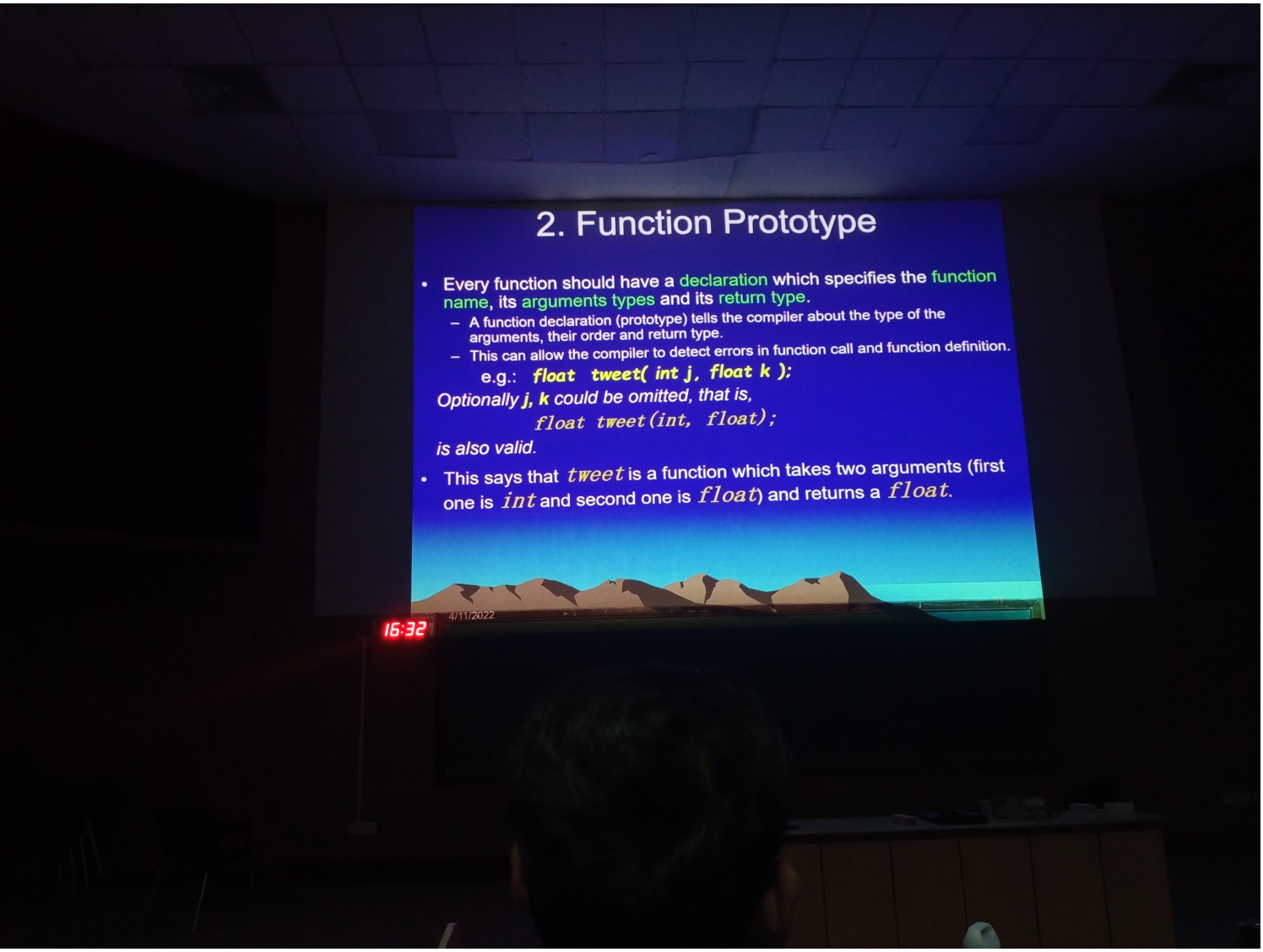

One more example
#include<stdio.h>
int square(int y); /* function prototype */
int main( )
{
int x;
for (x = 1; x <= 10; x++)
printf("%d ", square(x));
printf("\n");
return 0;
}
int square(int y)
{
return y * y;
}
16:20
4/11/2022

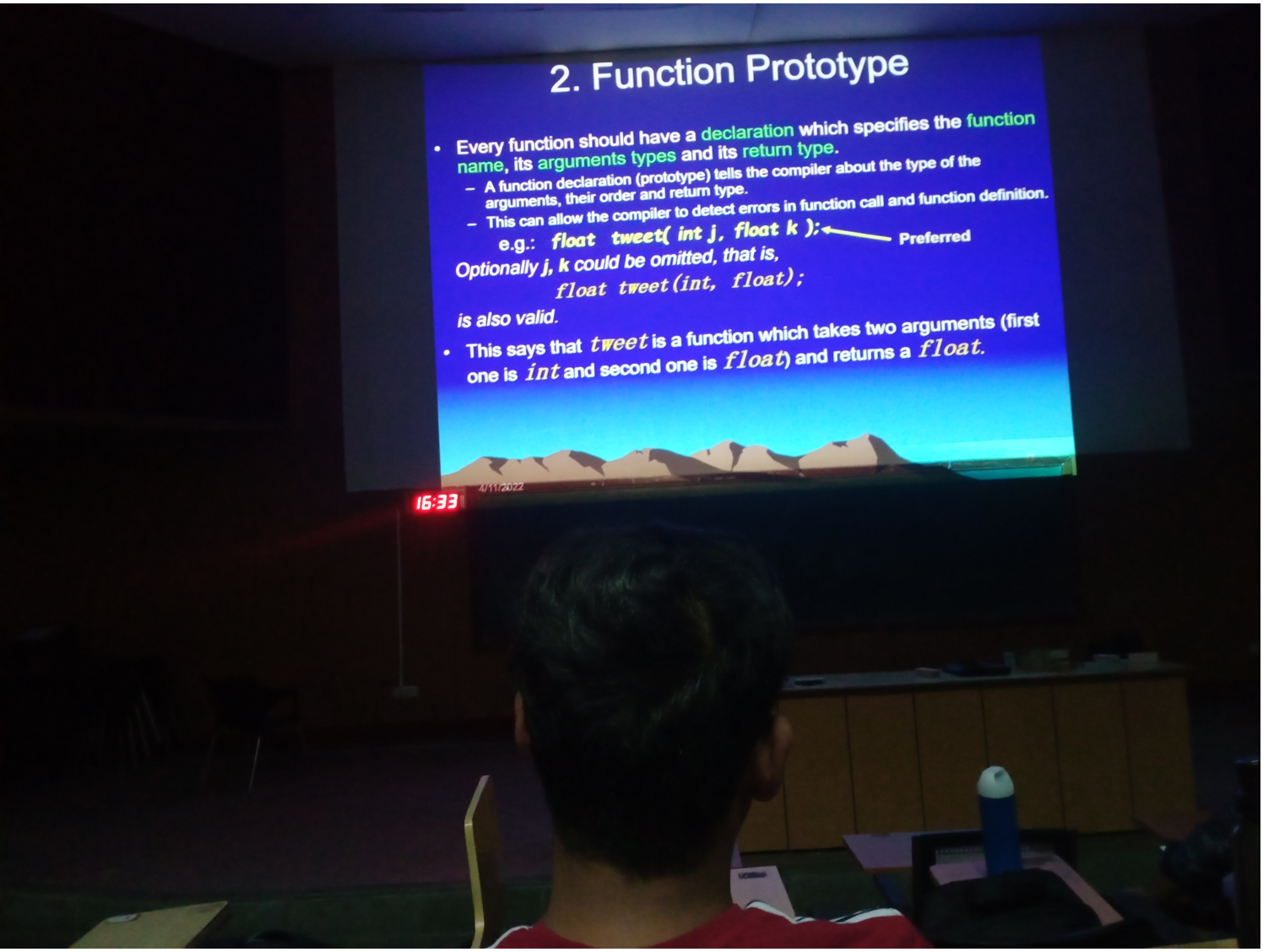

One more example
#include<stdio.h>
int square(int y); /* function prototype */
int main( )
{
int x;
for (x = 1; x <= 10; x++)
printf("%d ", square(x));
printf("\n");
return 0;
}
int square(int y)
{
return y * y;
}
main() is also a function.
It returns an int
If unspecified, then by default a function returns int
4/11/2022

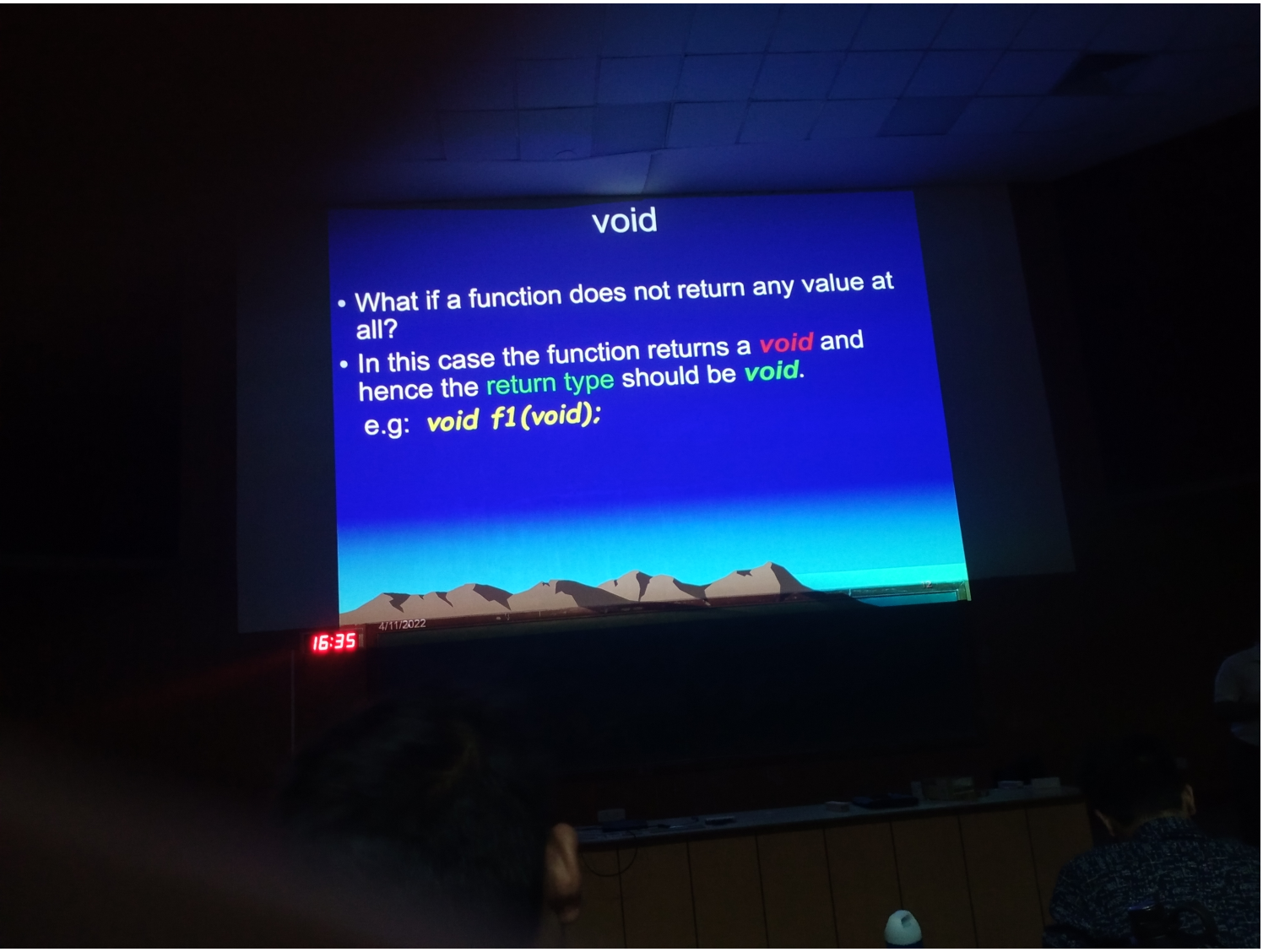

1. Control Flow
Execution starts with main() and when a function call like
z = func(exp1,exp2, ...); is encountered within main()
then:
Zzzzzz...
i. exp1, exp2, ... are evaluated
ii. The values of exp1, exp2, ... are passed to func( )
iii. main() is suspended temporarily
iv. func( ) is started with the supplied values
4/11/2022

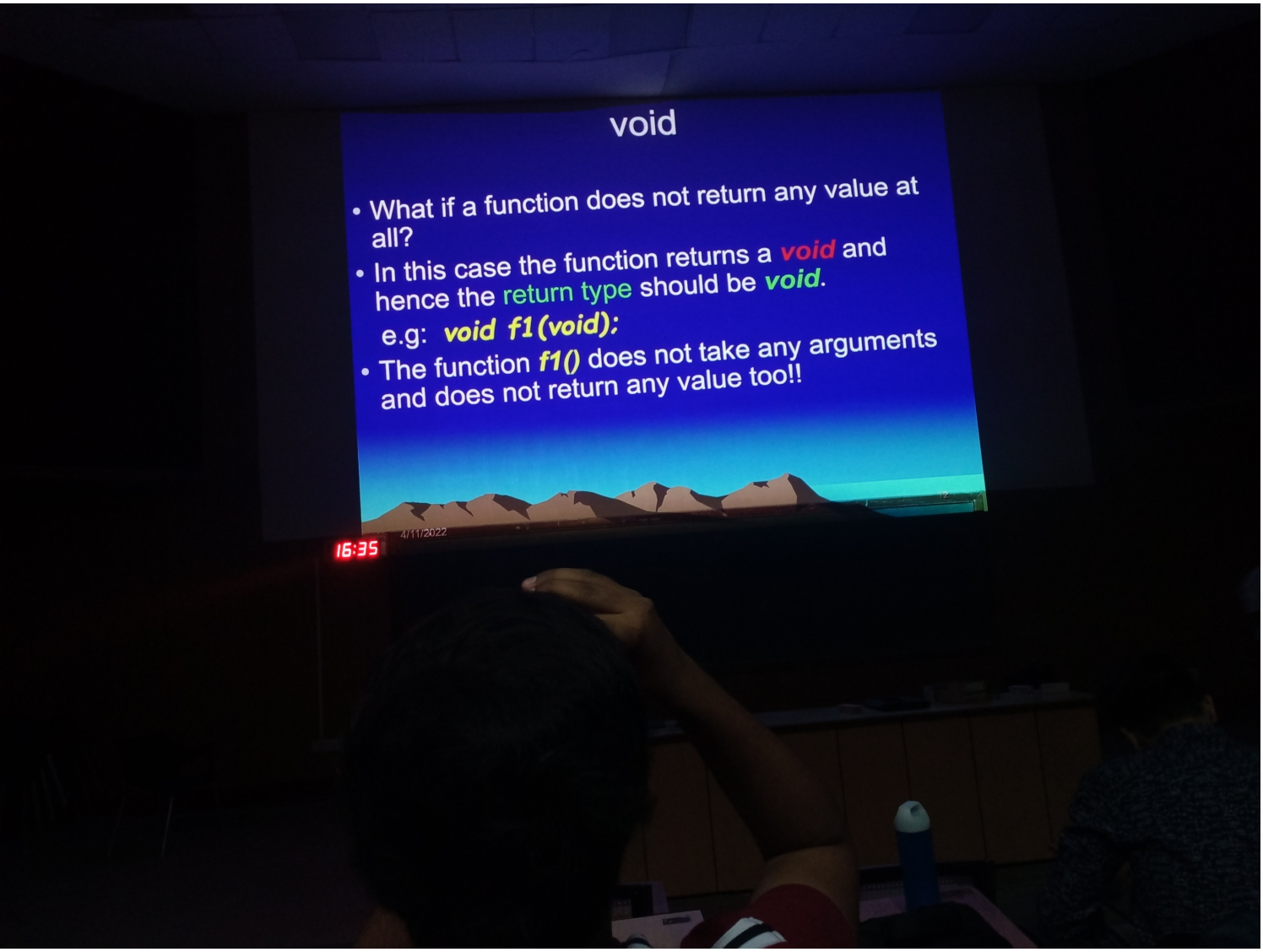

1. Control Flow
Execution starts with main() and when a function call like
z = func(exp1,exp2, ...); is encountered within main()
then:
Zzzzzz...
i. exp1, exp2, ... are evaluated
ii. The values of exp1, exp2, ... are passed to func( )
iii. main() is suspended temporarily
iv. func( ) is started with the supplied values
v. func( ) does the processing
vi. func( ) returns a value which is then assigned to z
16:21

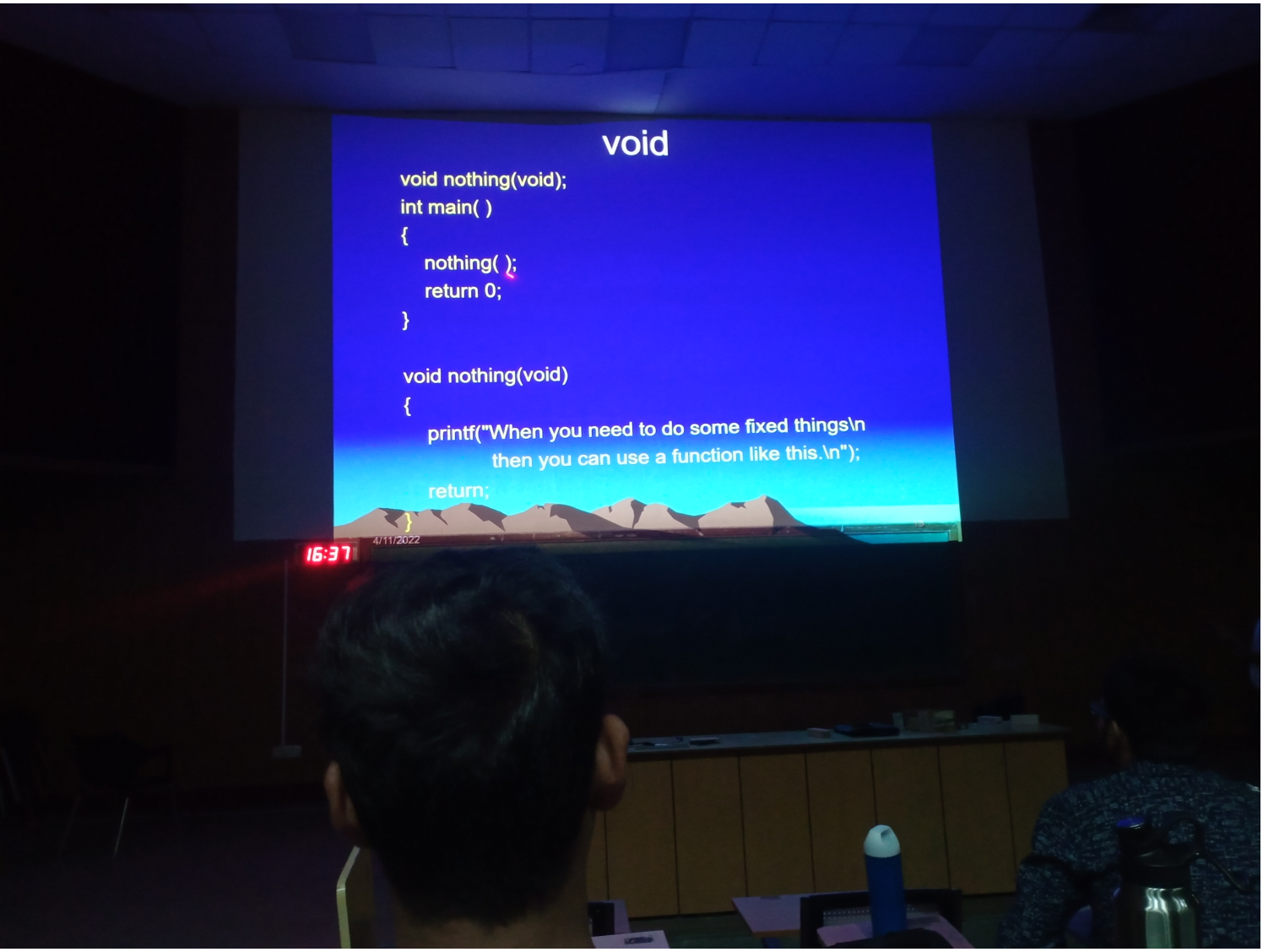

2. Function Prototype

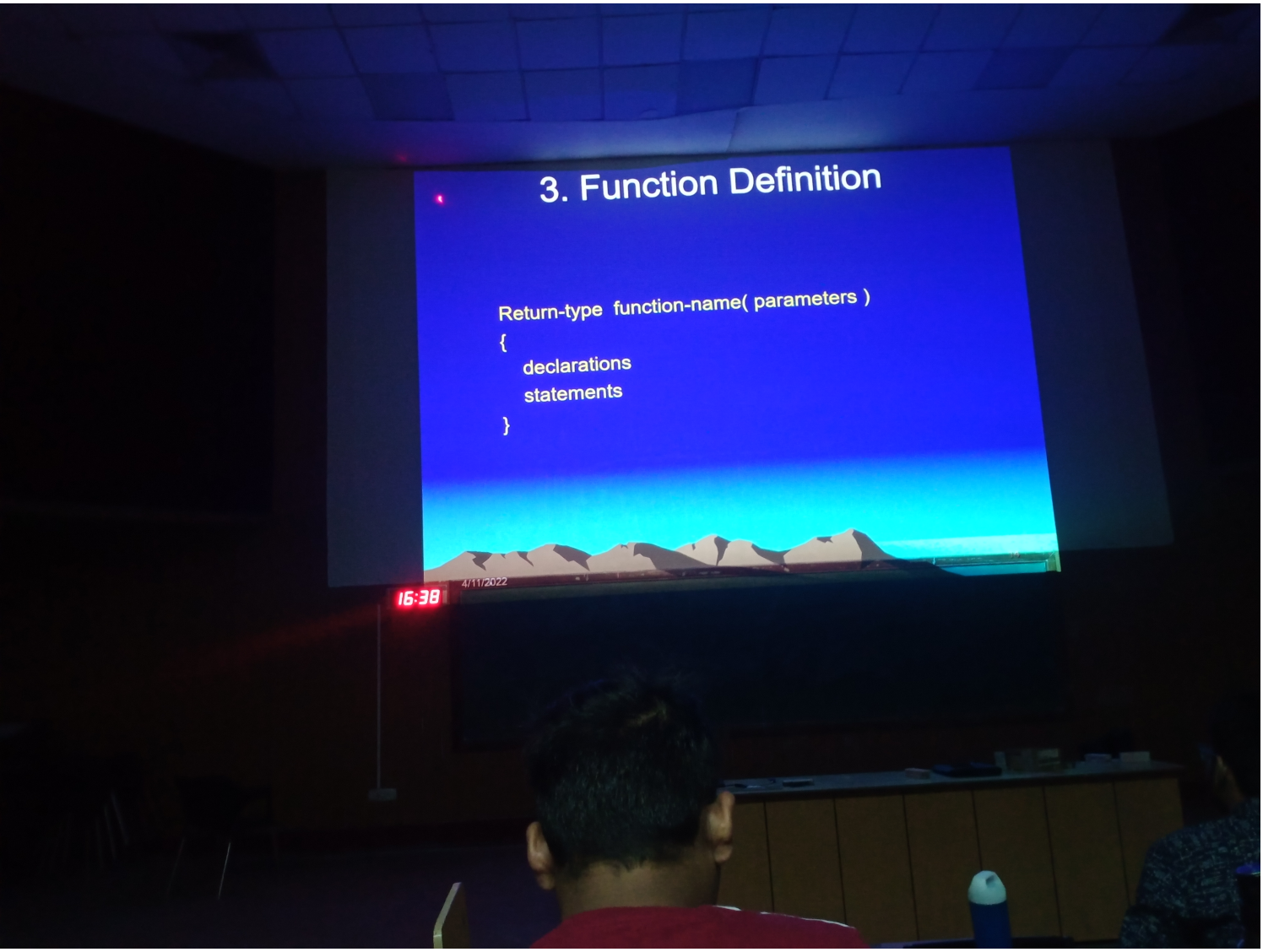

2. Function Prototype
• Every function should have a declaration which specifies the function name, its arguments types and its return type.
– A function declaration (prototype) tells the compiler about the type of the arguments, their order and return type.
– This can allow the compiler to detect errors in function call and function definition.
e.g.: float tweet(int j, float k);
Optionally j, k could be omitted, that is,
float tweet(int, float);
is also valid.
• This says that tweet is a function which takes two arguments (first one is int and second one is float) and returns a float.
16:32
4/11/2022

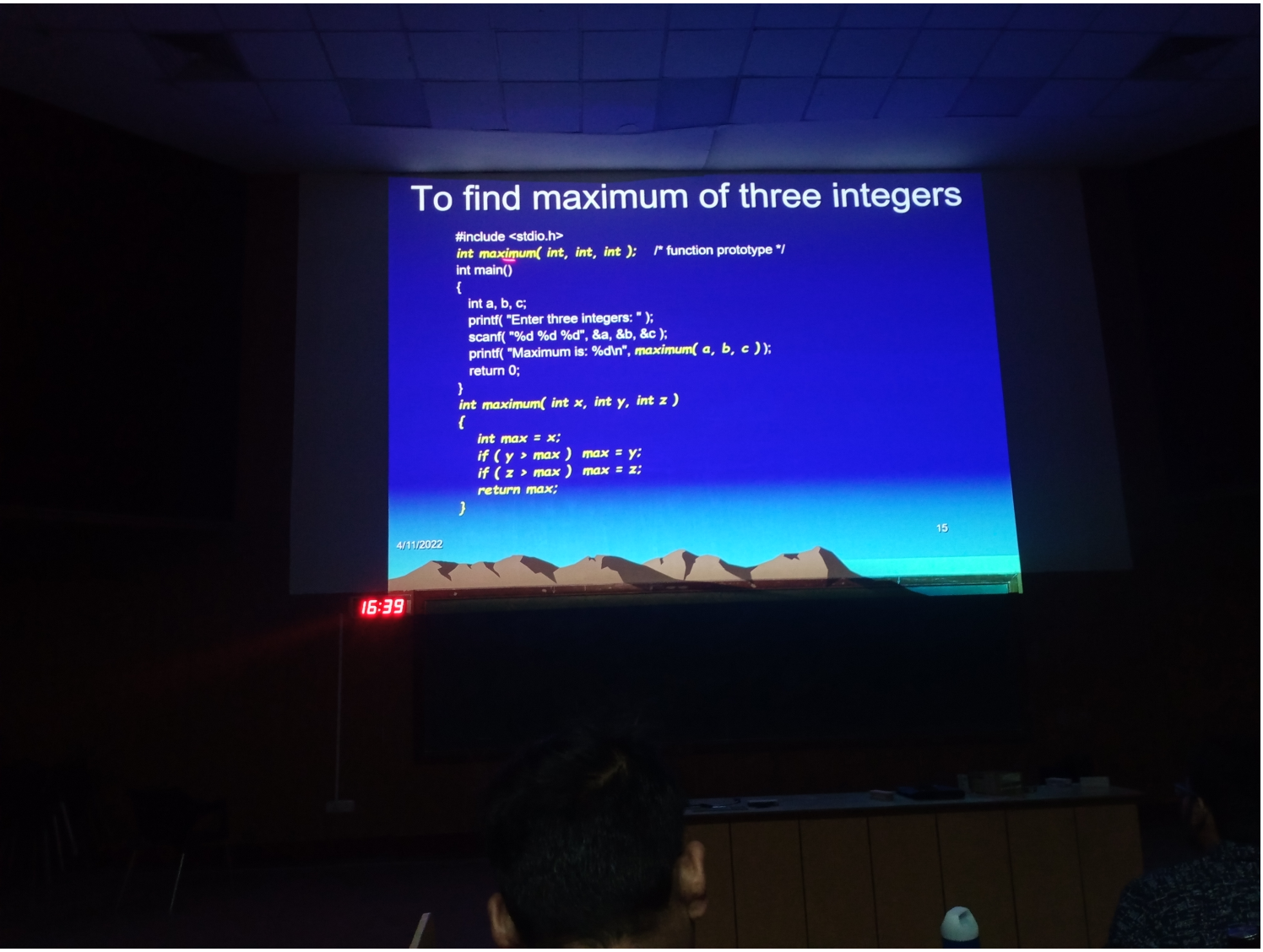

2. Function Prototype
Every function should have a declaration which specifies the function name, its arguments types and its return type.
A function declaration (prototype) tells the compiler about the type of the arguments, their order and return type.
This can allow the compiler to detect errors in function call and function definition.
e.g.: float tweet(int j, float k);
Preferred
Optionally j, k could be omitted, that is,
float tweet(int, float);
is also valid.
This says that tweet is a function which takes two arguments (first one is int and second one is float) and returns a float.
16:33
4/11/2022

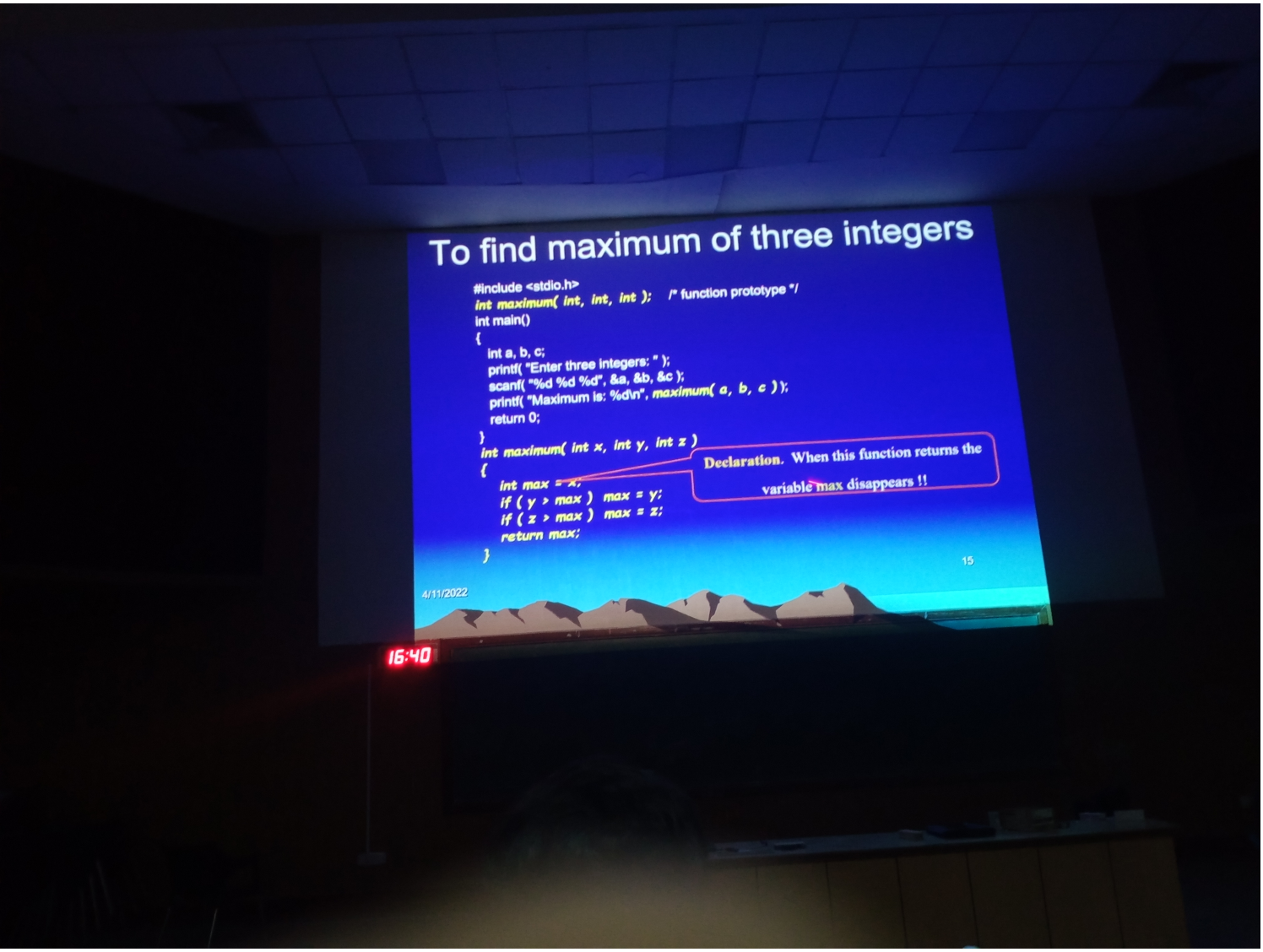

void
• What if a function does not return any value at all?
• In this case the function returns a void and hence the return type should be void.
e.g: void f1(void);
16:35
4/11/2022

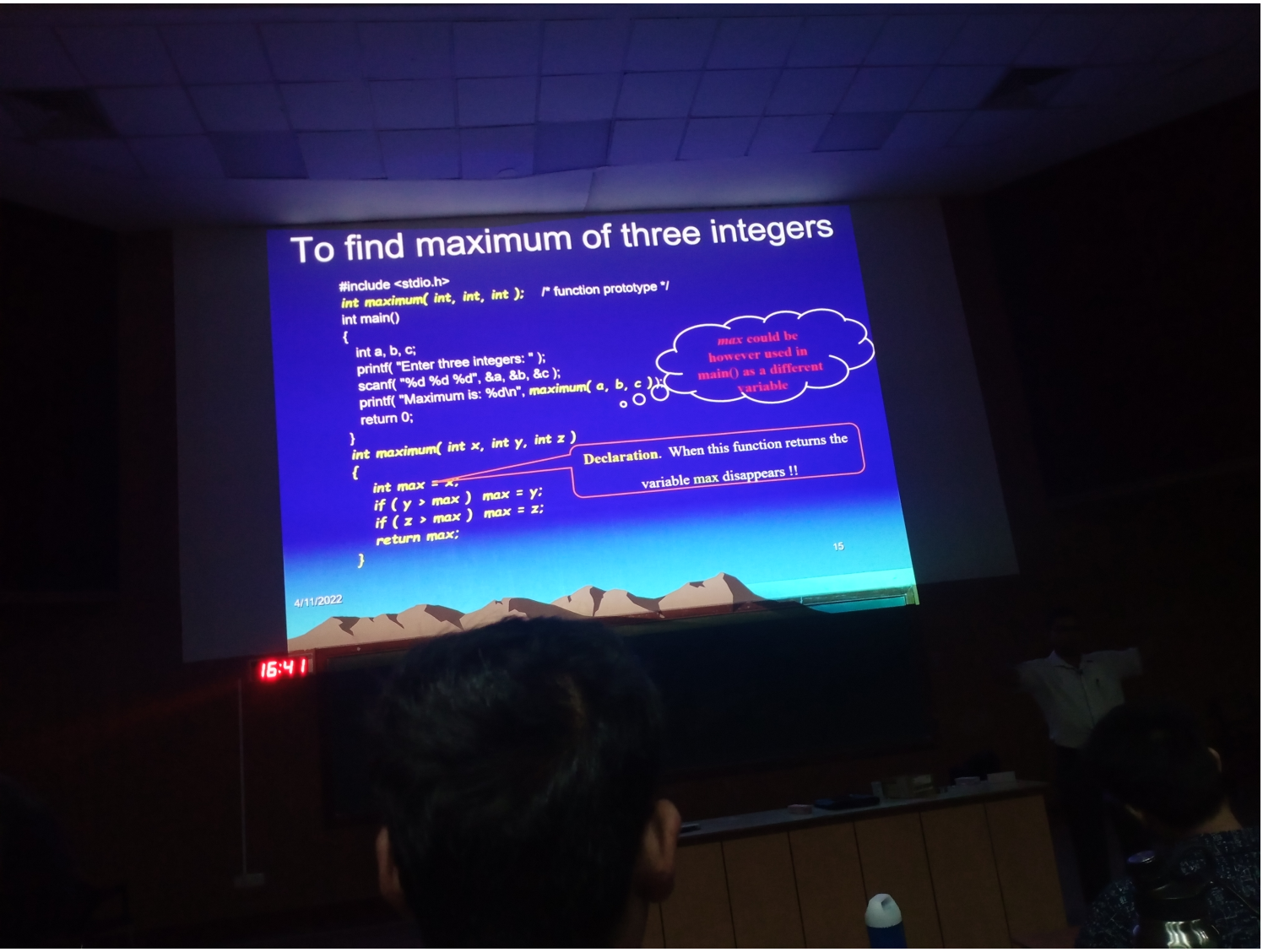

void
• What if a function does not return any value at all?
• In this case the function returns a void and hence the return type should be void.
e.g: void f1(void);
• The function f1() does not take any arguments and does not return any value too!!
16:35
4/11/2022

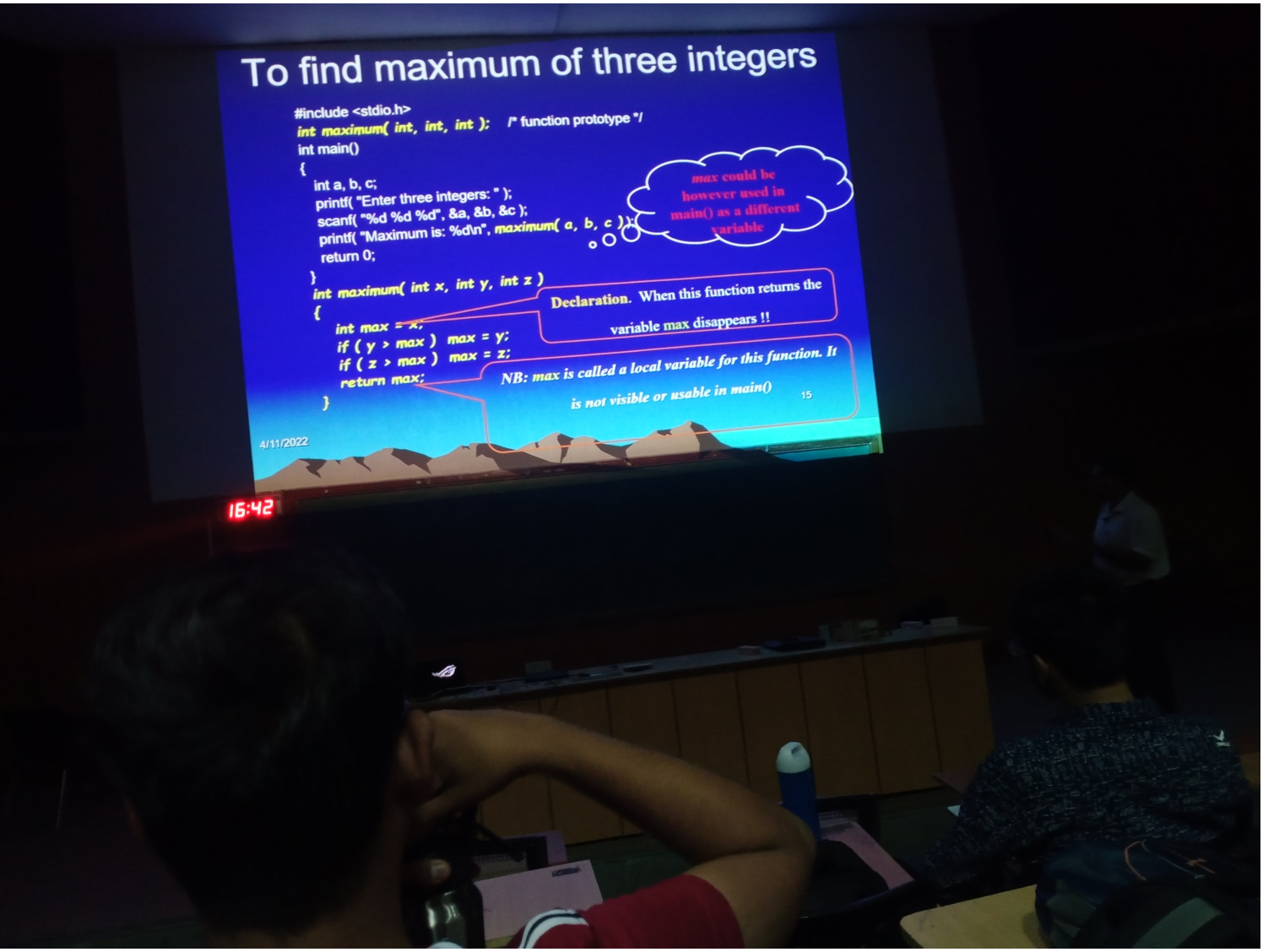

void
void nothing(void);
int main( )
{
nothing( );
return 0;
}
void nothing(void)
{
printf("When you need to do some fixed things\n
then you can use a function like this.\n");
return;
}
16:37
4/11/2022

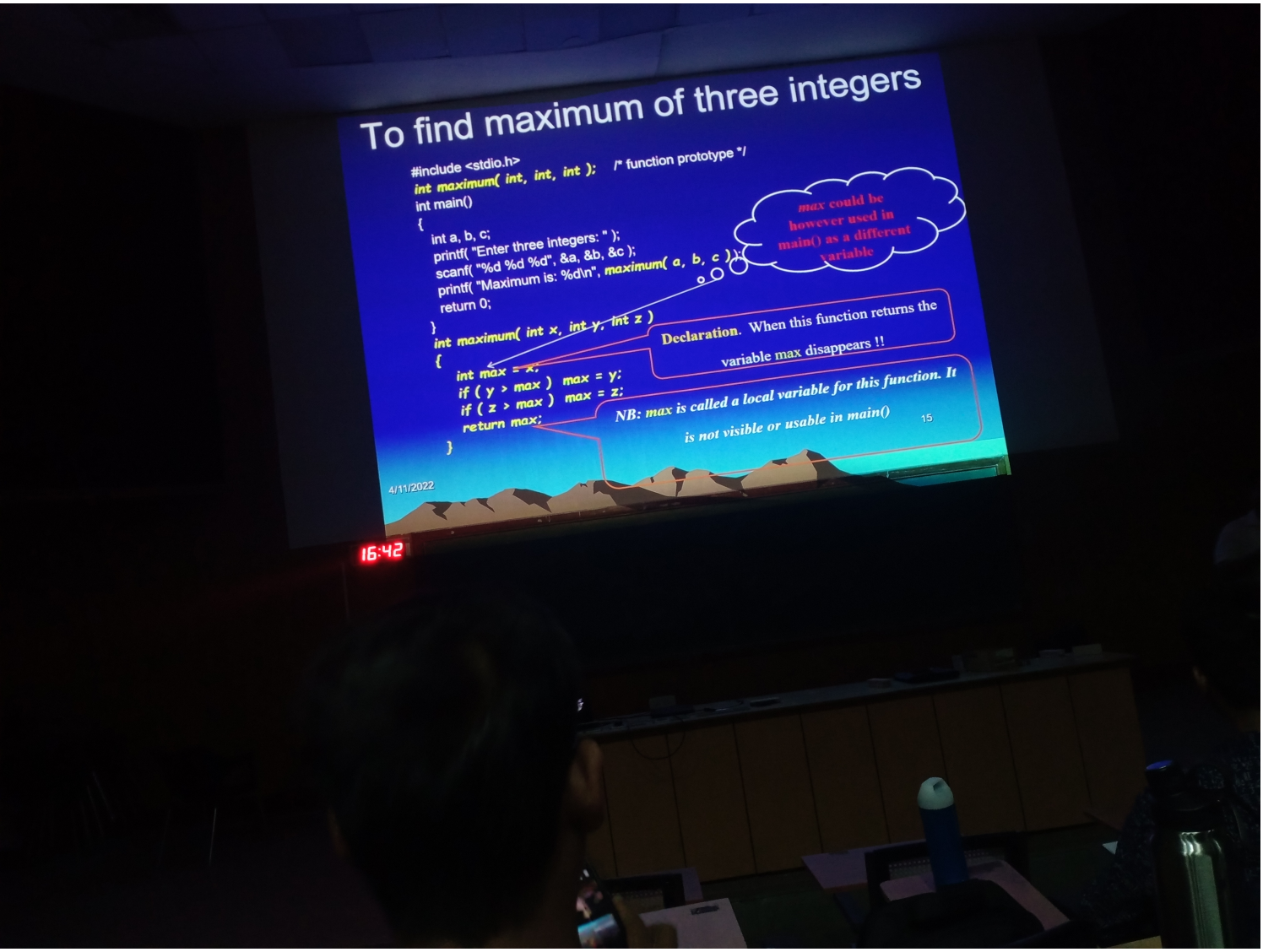

3. Function Definition
Return-type function-name(parameters)
{
declarations
statements
}
16:38
4/11/2022

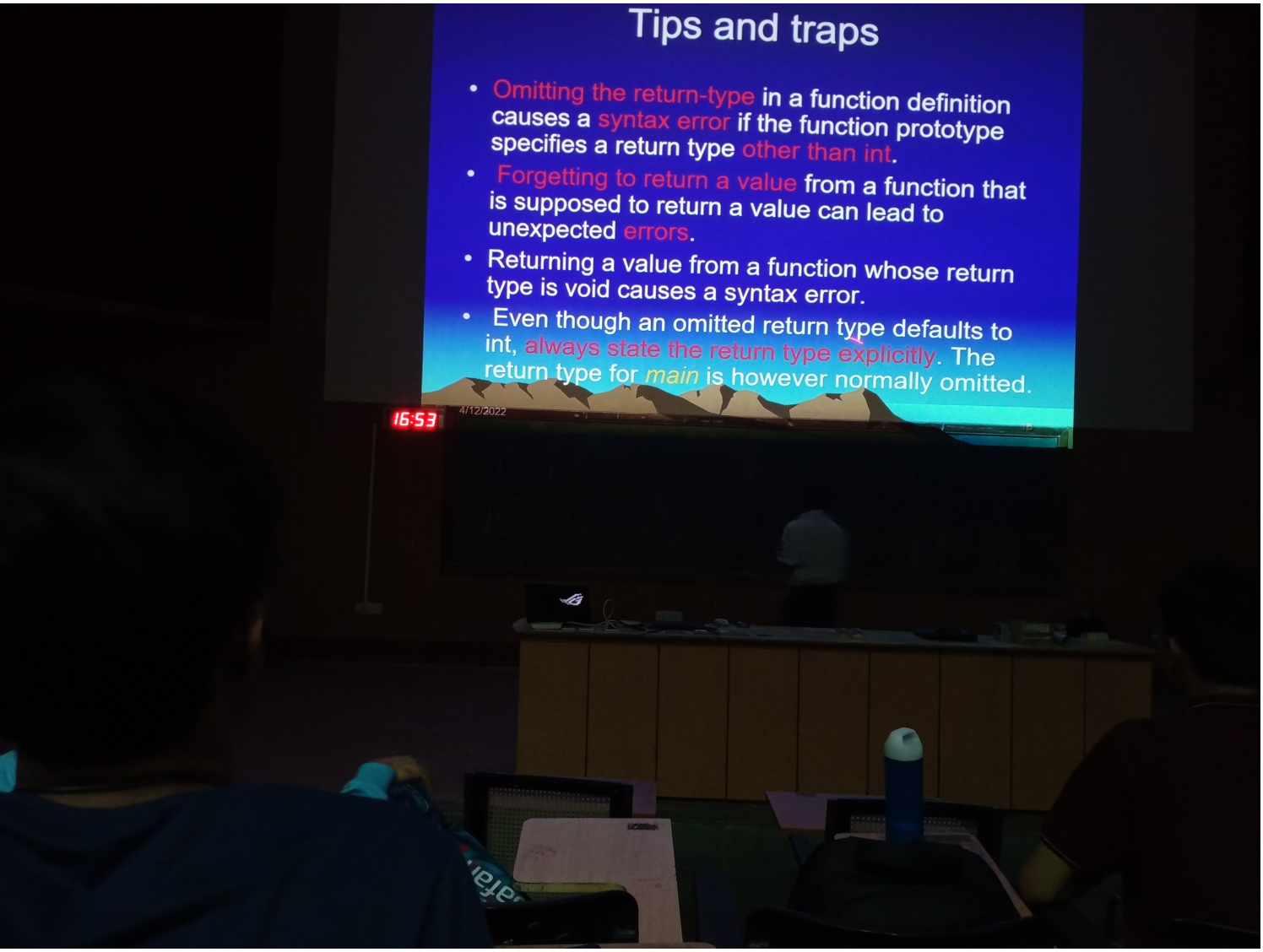

To find maximum of three integers
#include <stdio.h>
int maximum(int, int, int); /* function prototype */
int main()
{
int a, b, c;
printf("Enter three integers: ");
scanf("%d %d %d", &a, &b, &c);
printf("Maximum is: %d\n", maximum(a, b, c));
return 0;
}
int maximum(int x, int y, int z)
{
int max = x;
if (y > max) max = y;
if (z > max) max = z;
return max;
}
4/11/2022
15
16:39

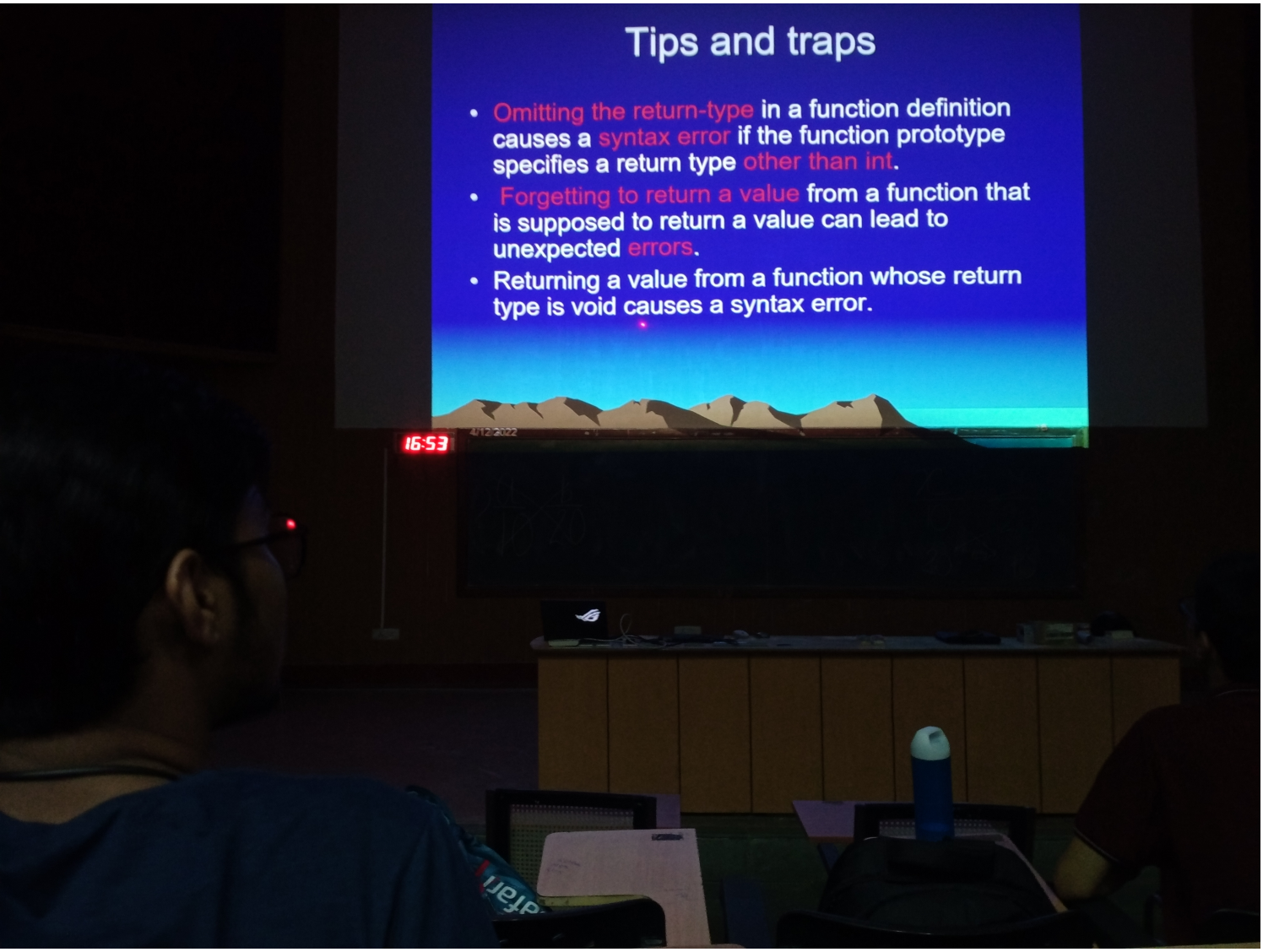

To find maximum of three integers
#include <stdio.h>
int maximum(int, int, int); /* function prototype */
int main()
{
int a, b, c;
printf("Enter three integers: ");
scanf("%d %d %d", &a, &b, &c);
printf("Maximum is: %d\n", maximum(a, b, c));
return 0;
}
int maximum(int x, int y, int z)
{
int max = x;
if (y > max) max = y;
if (z > max) max = z;
return max;
}
Declaration. When this function returns the variable max disappears !!
15
4/11/2022
16:40

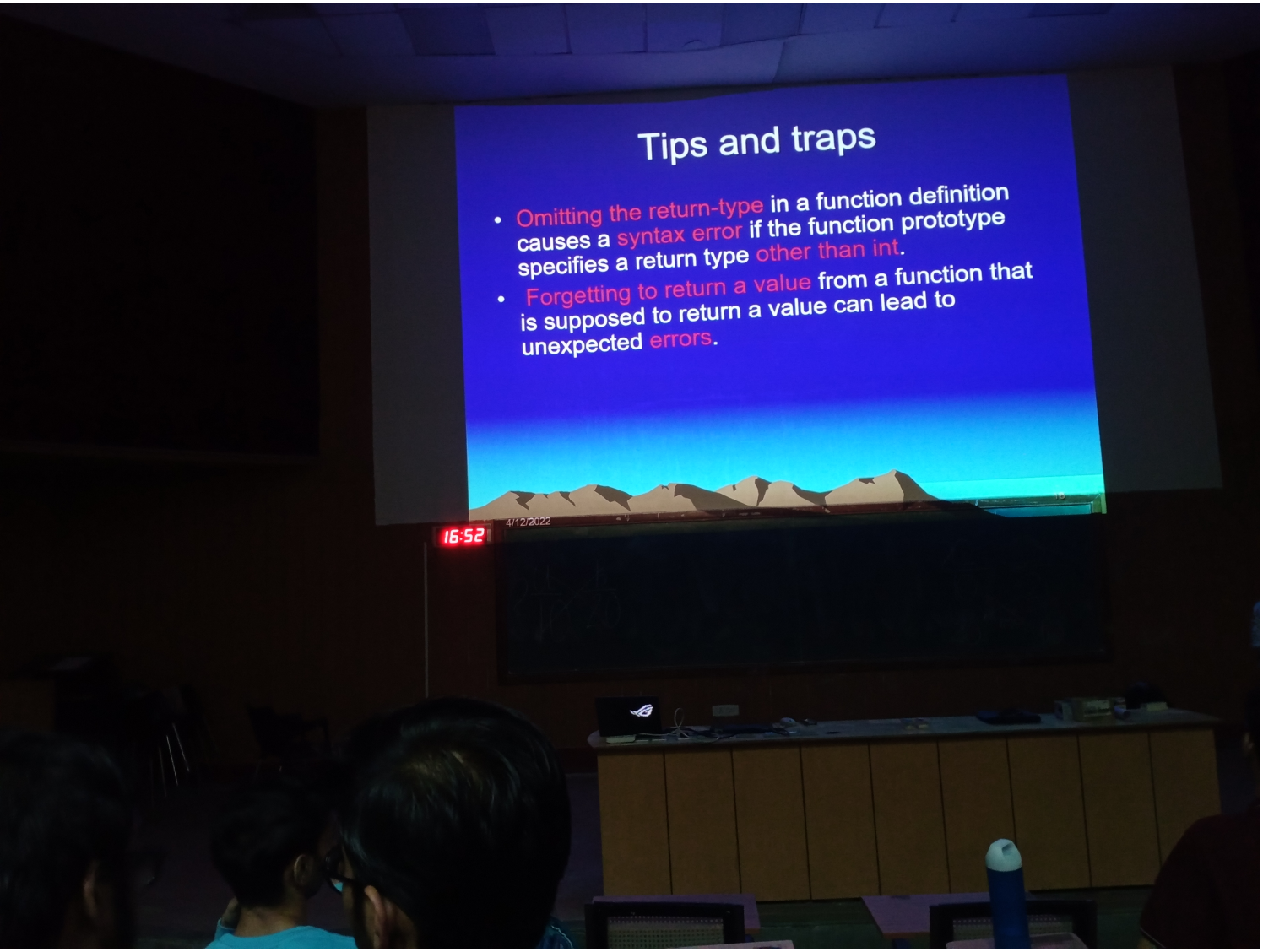

To find maximum of three integers
#include <stdio.h>
int maximum(int, int, int); /* function prototype */
int main()
{
int a, b, c;
printf("Enter three integers: ");
scanf("%d %d %d", &a, &b, &c);
printf("Maximum is: %d\n", maximum(a, b, c));
return 0;
}
int maximum(int x, int y, int z)
{
int max = x;
if (y > max) max = y;
if (z > max) max = z;
return max;
}
max could be however used in main() as a different variable
Declaration. When this function returns the variable max disappears !!
15
4/11/2022

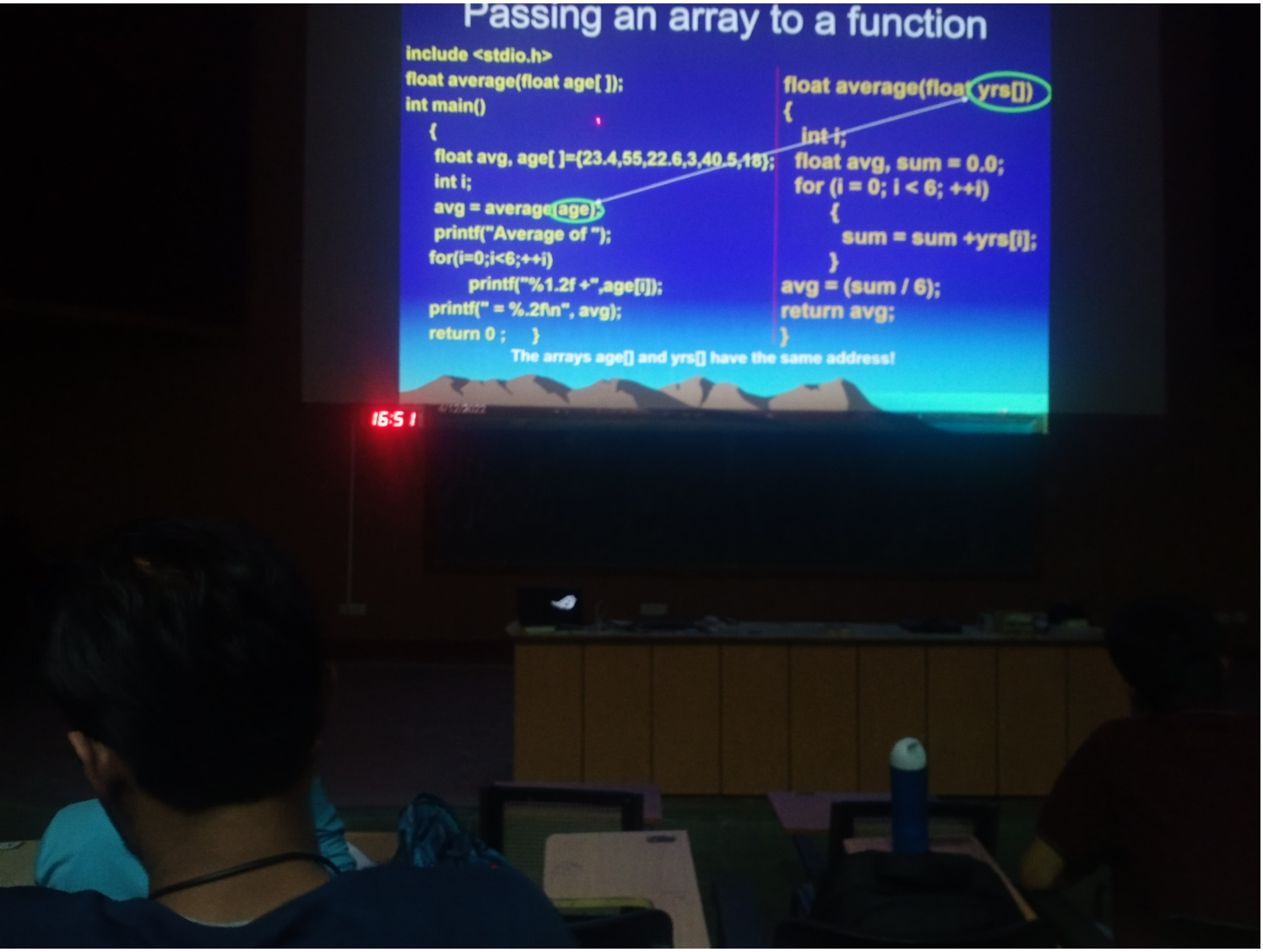

To find maximum of three integers
#include <stdio.h>
int maximum(int, int, int); /* function prototype */
int main()
{
int a, b, c;
printf("Enter three integers: ");
scanf("%d %d %d", &a, &b, &c);
printf("Maximum is: %d\n", maximum(a, b, c));
return 0;
}
int maximum(int x, int y, int z)
{
int max = x;
if (y > max) max = y;
if (z > max) max = z;
return max;
}
max could be however used in main() as a different variable
Declaration. When this function returns the variable max disappears !!
NB: max is called a local variable for this function. It is not visible or usable in main()
15
4/11/2022
16:42

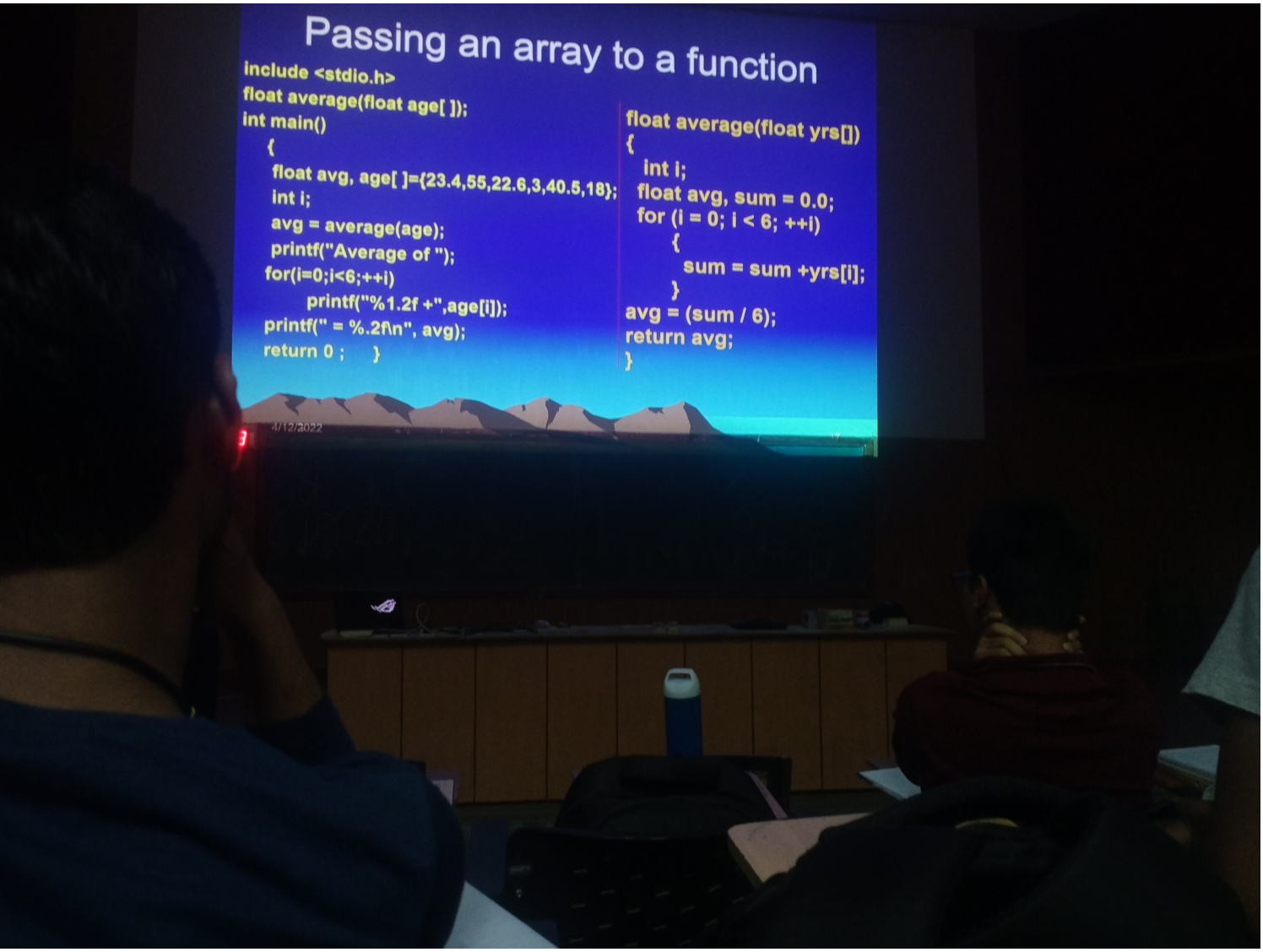

To find maximum of three integers
#include <stdio.h>
int maximum(int, int, int); /* function prototype */
int main()
{
int a, b, c;
printf("Enter three integers: ");
scanf("%d %d %d", &a, &b, &c);
printf("Maximum is: %d\n", maximum(a, b, c));
return 0;
}
int maximum(int x, int y, int z)
{
int max = x;
if (y > max) max = y;
if (z > max) max = z;
return max;
}
max could be however used in main() as a different variable
Declaration. When this function returns the variable max disappears !!
NB: max is called a local variable for this function. It is not visible or usable in main()
15
4/11/2022
16:42

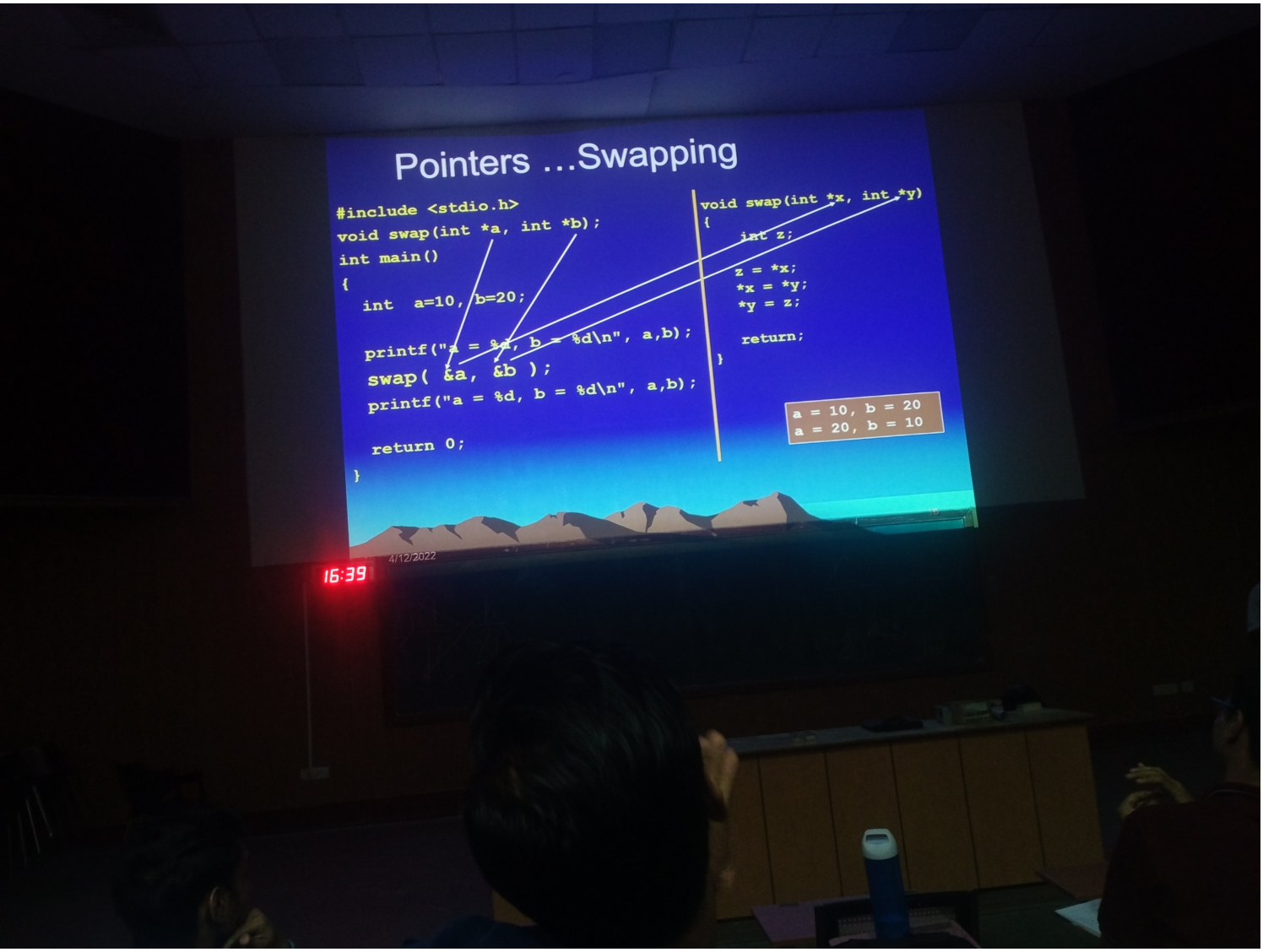

Tips and traps
• Omitting the return-type in a function definition causes a syntax error if the function prototype specifies a return type other than int.
• Forgetting to return a value from a function that is supposed to return a value can lead to unexpected errors.
• Returning a value from a function whose return type is void causes a syntax error.
• Even though an omitted return type defaults to int, always state the return type explicitly. The return type for main is however normally omitted.
16:53
4/12/2022

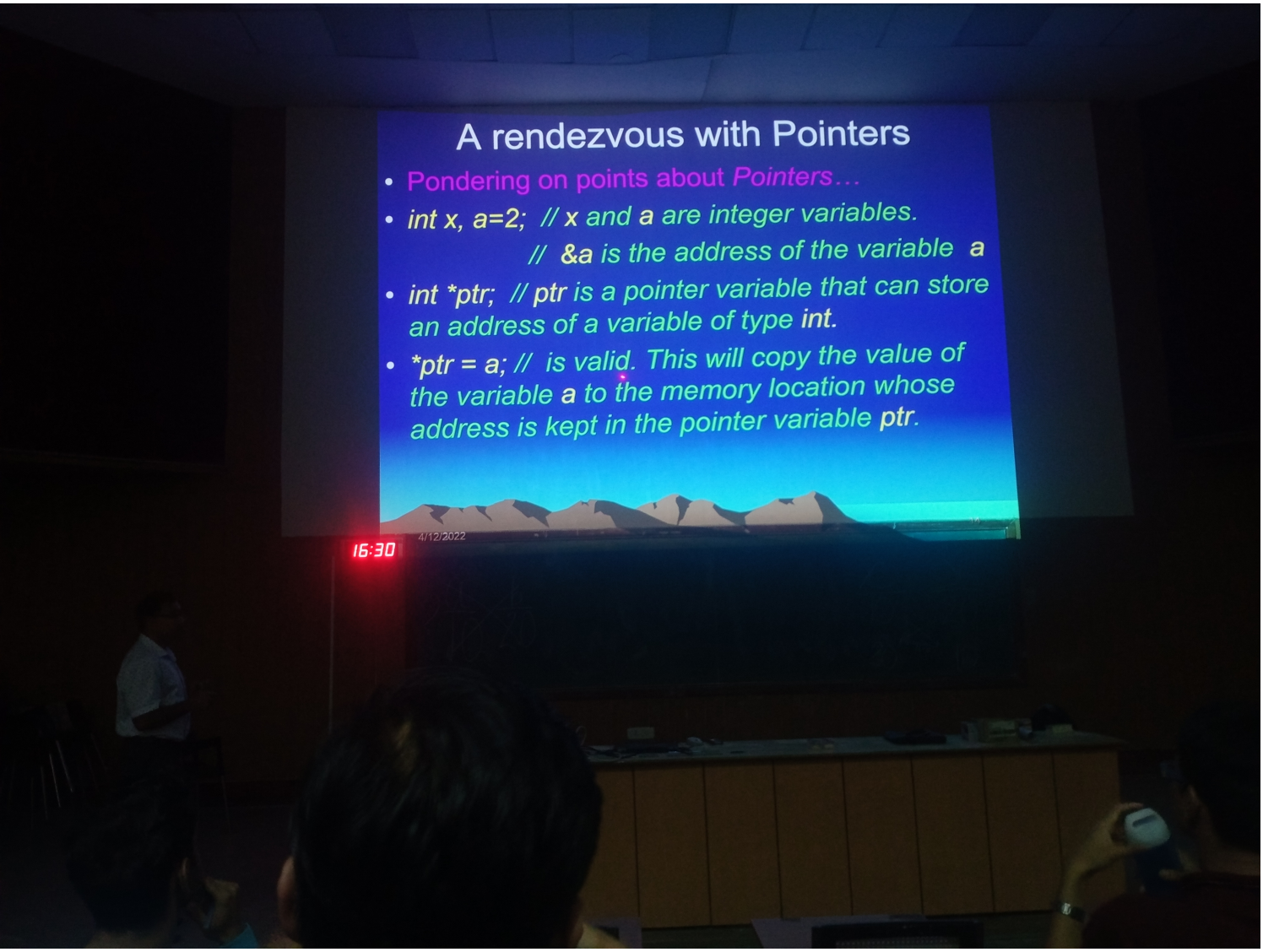

Tips and traps
Omitting the return-type in a function definition causes a syntax error if the function prototype specifies a return type other than int.
Forgetting to return a value from a function that is supposed to return a value can lead to unexpected errors.
Returning a value from a function whose return type is void causes a syntax error.
16:53
4/12/2022

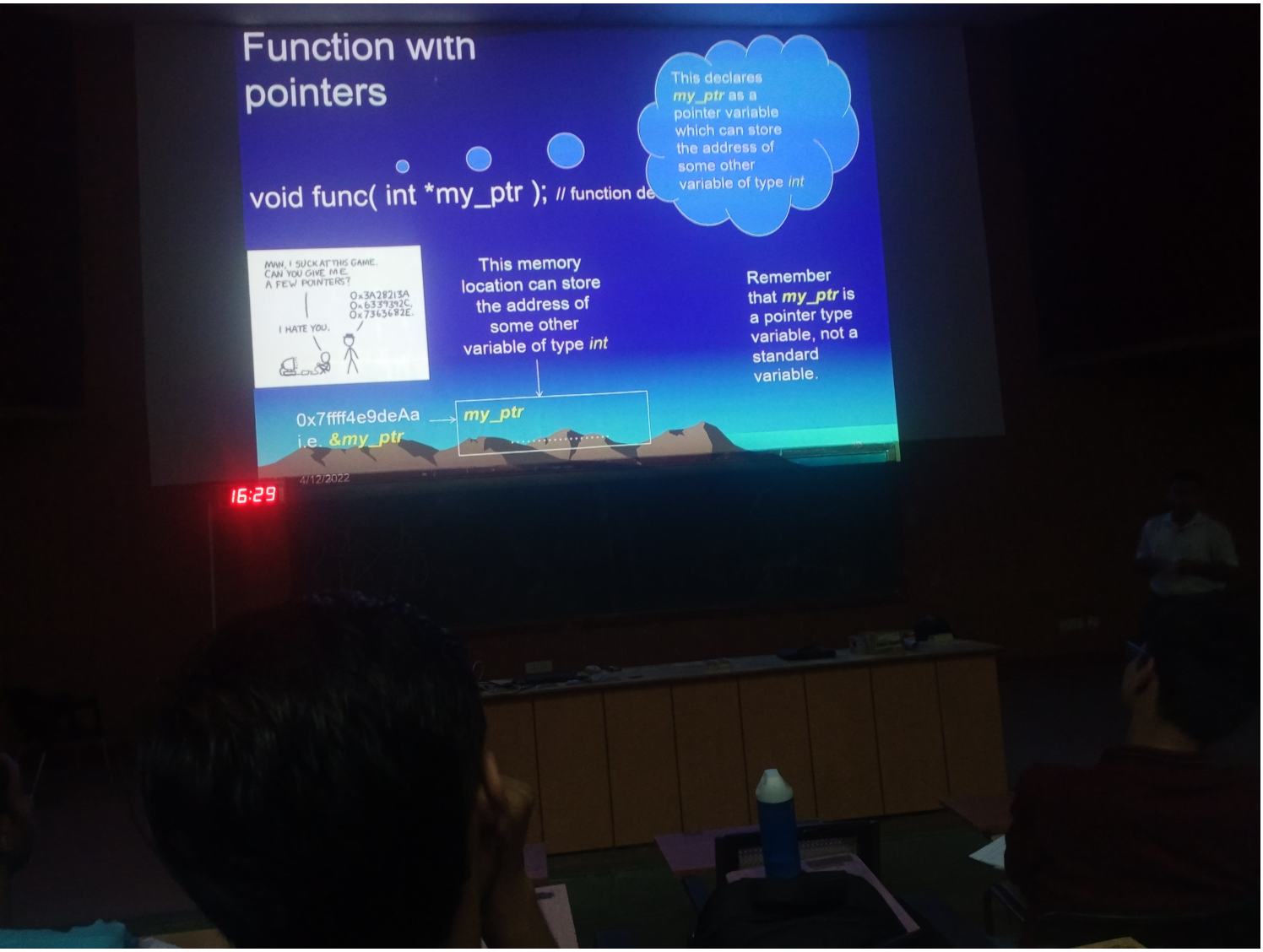

Tips and traps
• Omitting the return-type in a function definition causes a syntax error if the function prototype specifies a return type other than int.
• Forgetting to return a value from a function that is supposed to return a value can lead to unexpected errors.
16:52
4/12/2022

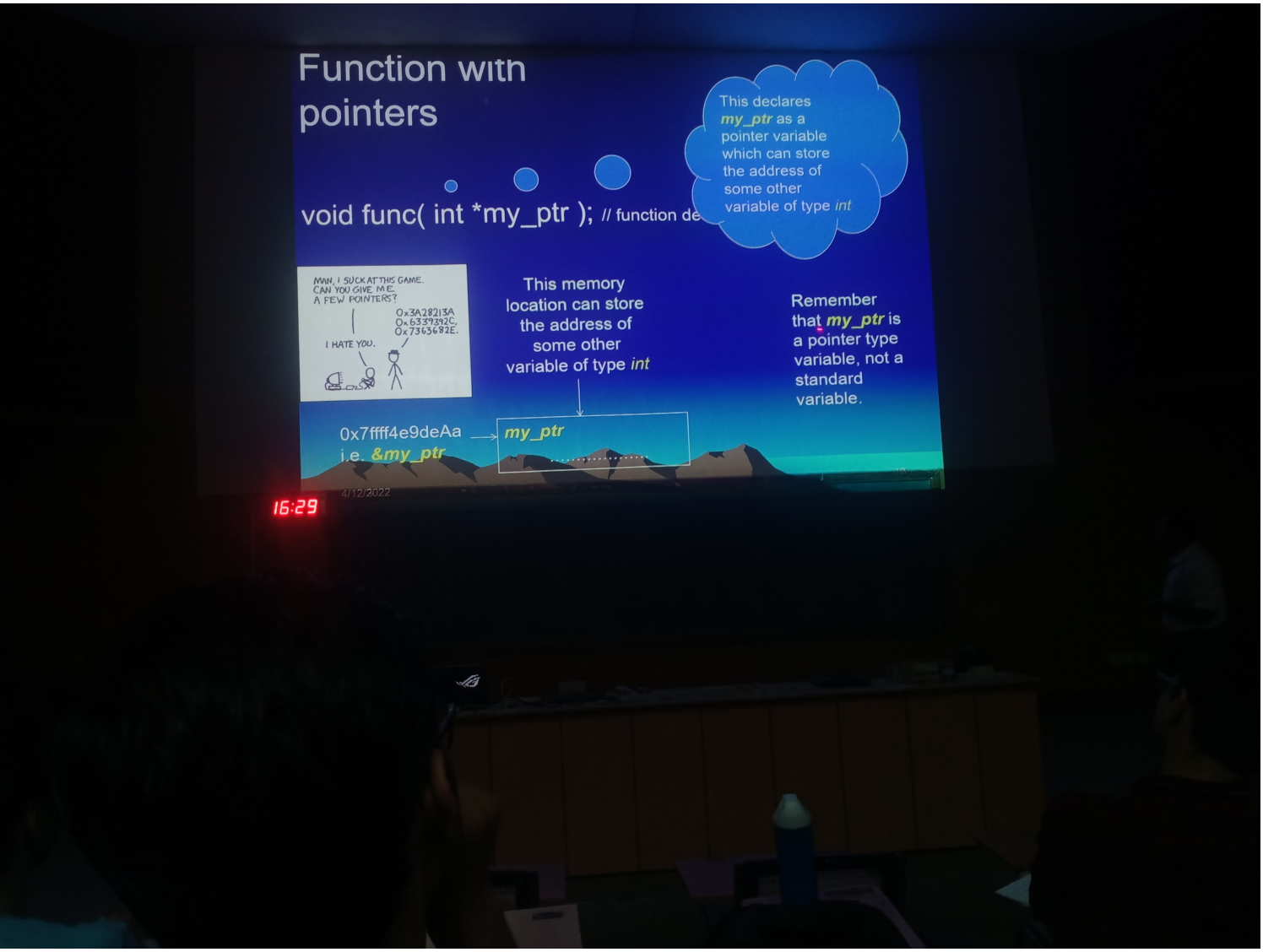

Passing an array to a function
include <stdio.h>
float average(float age[ ]);
int main()
{
float avg, age[ ]={23.4,55,22.6,3,40.5,18};
int i;
avg = average(age);
printf("Average of ");
for(i=0;i<6;++i)
printf("%1.2f +",age[i]);
printf(" = %.2f\n", avg);
return 0 ; }
float average(float yrs[])
{
int i;
float avg, sum = 0.0;
for (i = 0; i < 6; ++i)
{
sum = sum +yrs[i];
}
avg = (sum / 6);
return avg;
}
The arrays age[] and yrs[] have the same address!

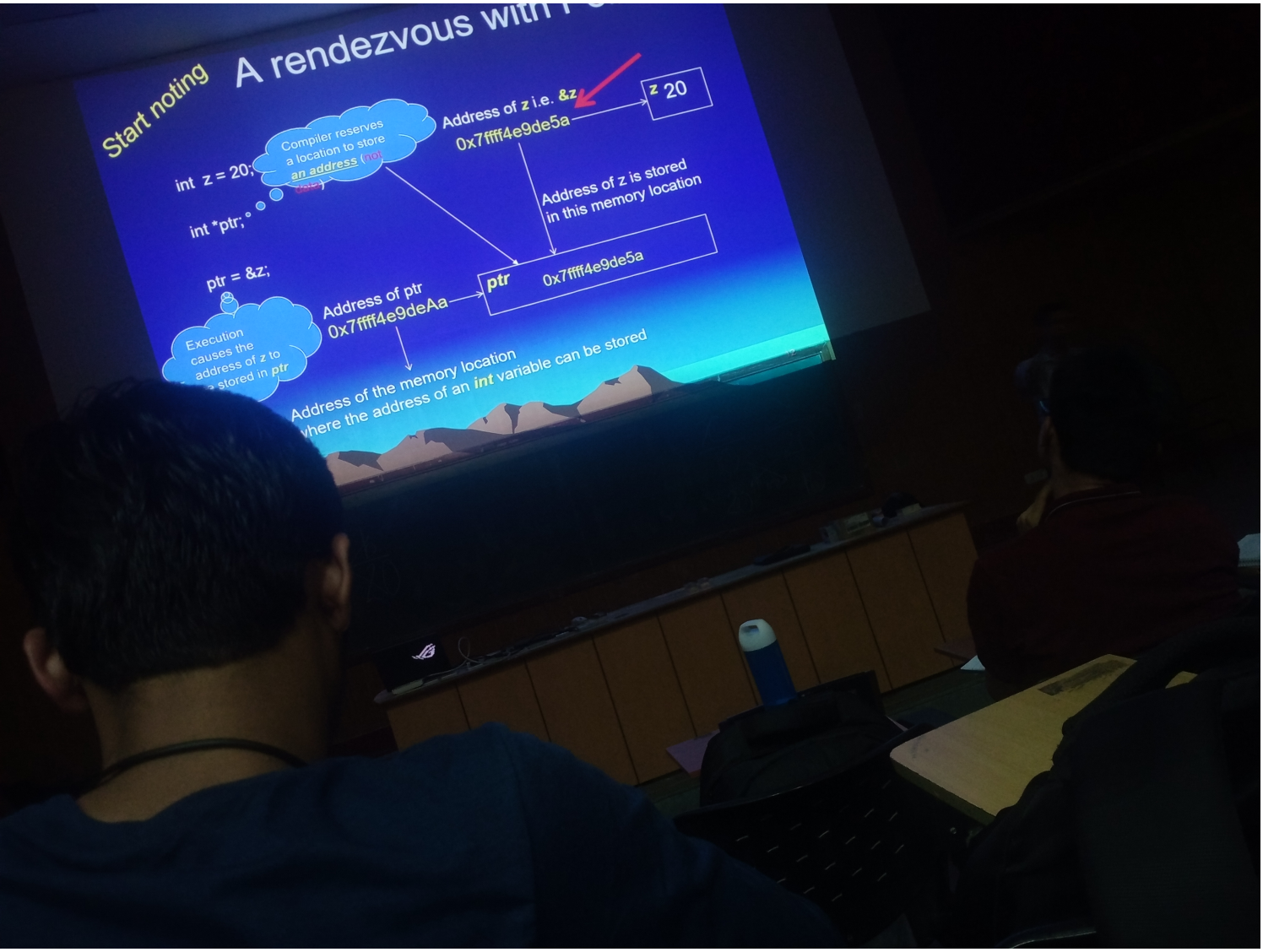

Passing an array to a function
include <stdio.h>
float average(float age[ ]);
int main()
{
float avg, age[ ]={23.4,55,22.6,3,40.5,18};
int i;
avg = average(age);
printf("Average of ");
for(i=0;i<6;++i)
printf("%1.2f +",age[i]);
printf(" = %.2f\n", avg);
return 0 ; }
float average(float yrs[])
{
int i;
float avg, sum = 0.0;
for (i = 0; i < 6; ++i)
{
sum = sum +yrs[i];
}
avg = (sum / 6);
return avg;
}
4/12/2022

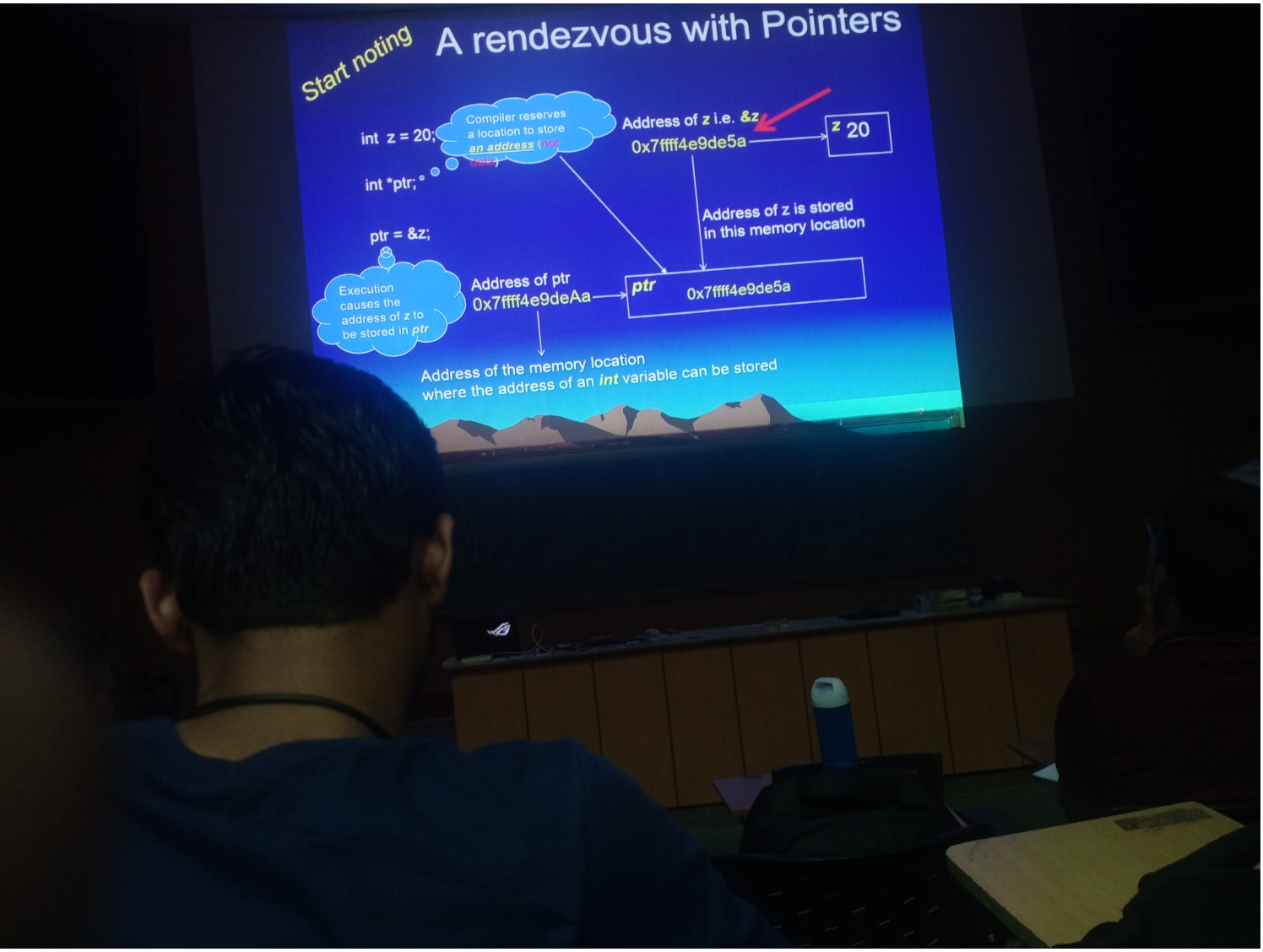

Pointers …Swapping
#include <stdio.h>
void swap(int *a, int *b);
int main()
{
int a=10, b=20;
printf("a = %d, b = %d\n", a,b);
swap(&a, &b);
printf("a = %d, b = %d\n", a,b);
return 0;
}
void swap(int *x, int *y)
{
int z;
z = *x;
*x = *y;
*y = z;
return;
}
a = 10, b = 20
a = 20, b = 10
4/12/2022
16:39

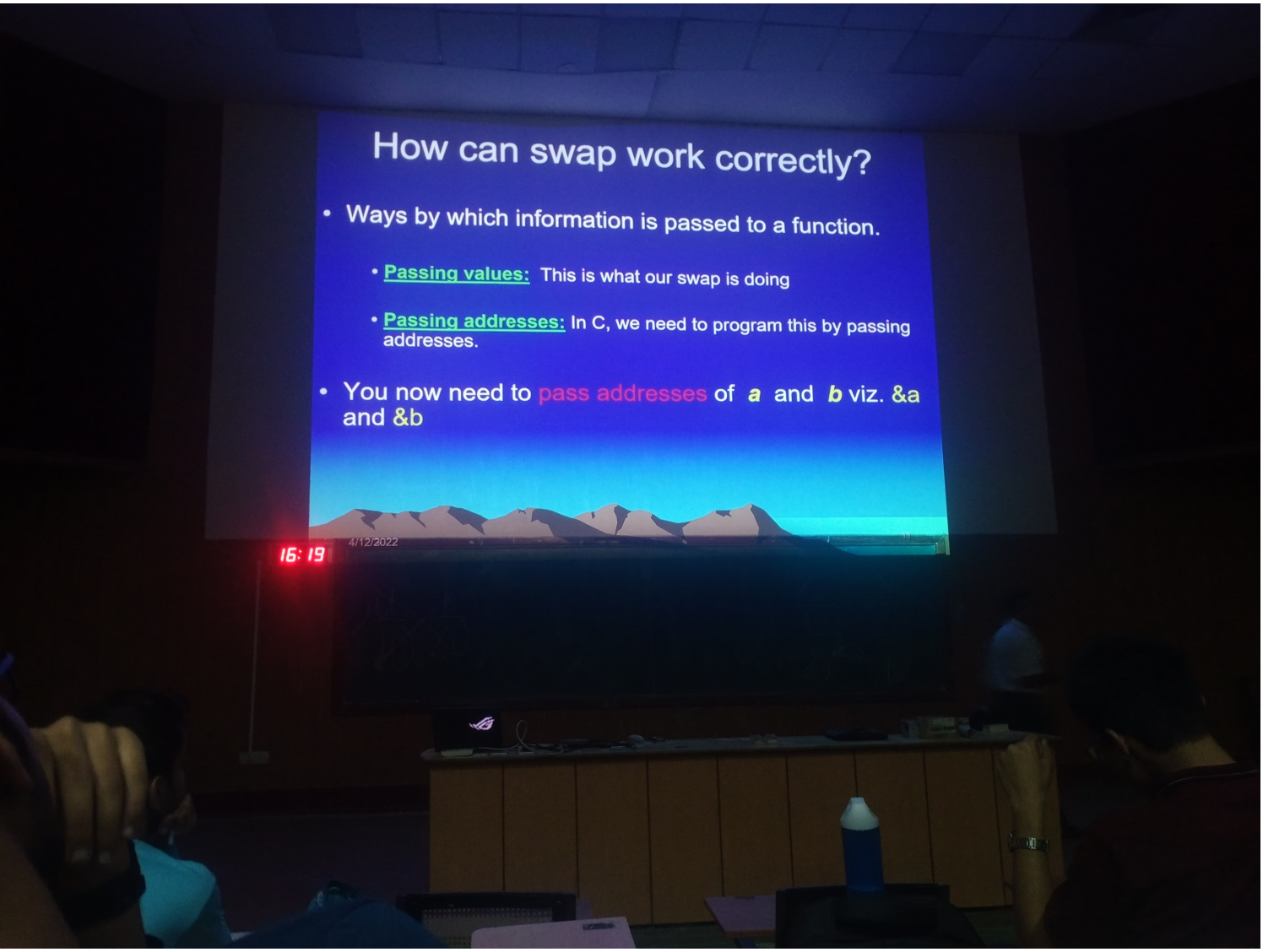

A rendezvous with Pointers
• Pondering on points about Pointers…
• int x, a=2; // x and a are integer variables.
// &a is the address of the variable a
• int *ptr; // ptr is a pointer variable that can store an address of a variable of type int.
• *ptr = a; // is valid. This will copy the value of the variable a to the memory location whose address is kept in the pointer variable ptr.
16:30
4/12/2022

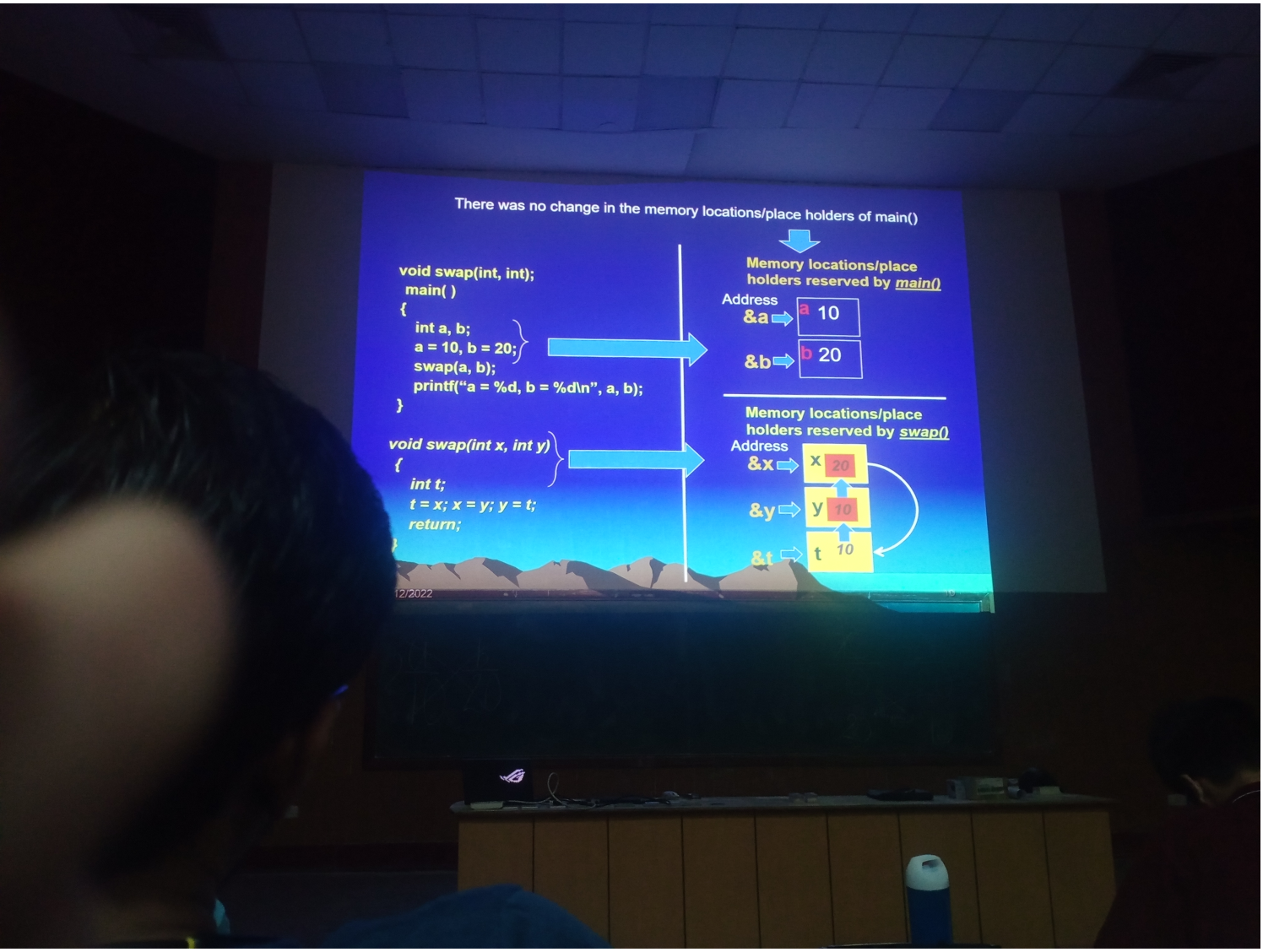

Function with pointers
void func(int *my_ptr); // function de
This declares my_ptr as a pointer variable which can store the address of some other variable of type int
MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x63373932.
0x73636B2E.
I HATE YOU.
This memory location can store the address of some other variable of type int
Remember that my_ptr is a pointer type variable, not a standard variable.
0x7ffff4e9deAa
i.e. &my_ptr
my_ptr
4/12/2022
16:29

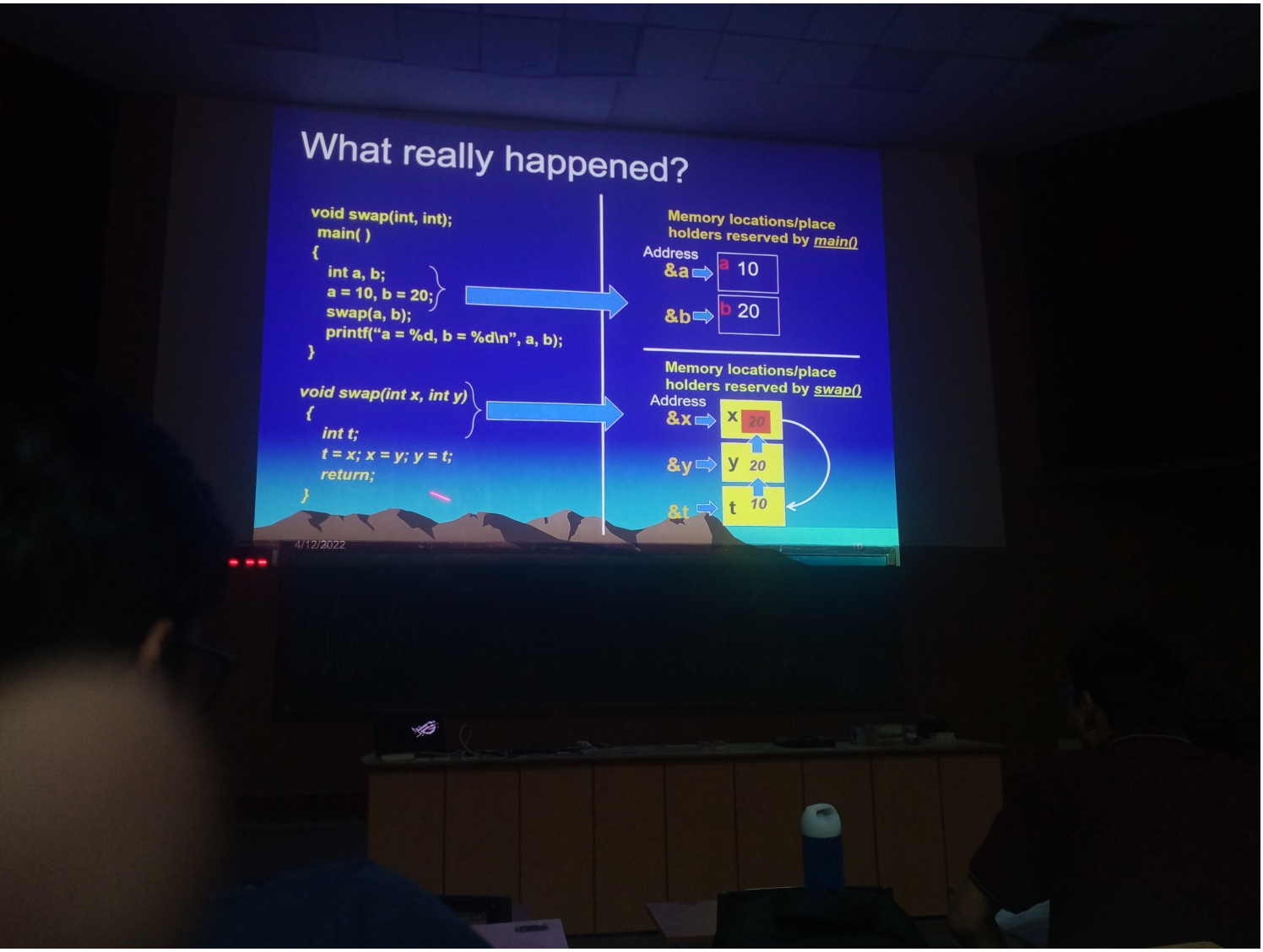

Function with pointers
void func(int *my_ptr); // function de
This declares my_ptr as a pointer variable which can store the address of some other variable of type int
MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x6339392C
0x7363682E
I HATE YOU.
This memory location can store the address of some other variable of type int
Remember that my_ptr is a pointer type variable, not a standard variable.
0x7ffff4e9deAa
i.e. &my_ptr
my_ptr
4/12/2022
16:29

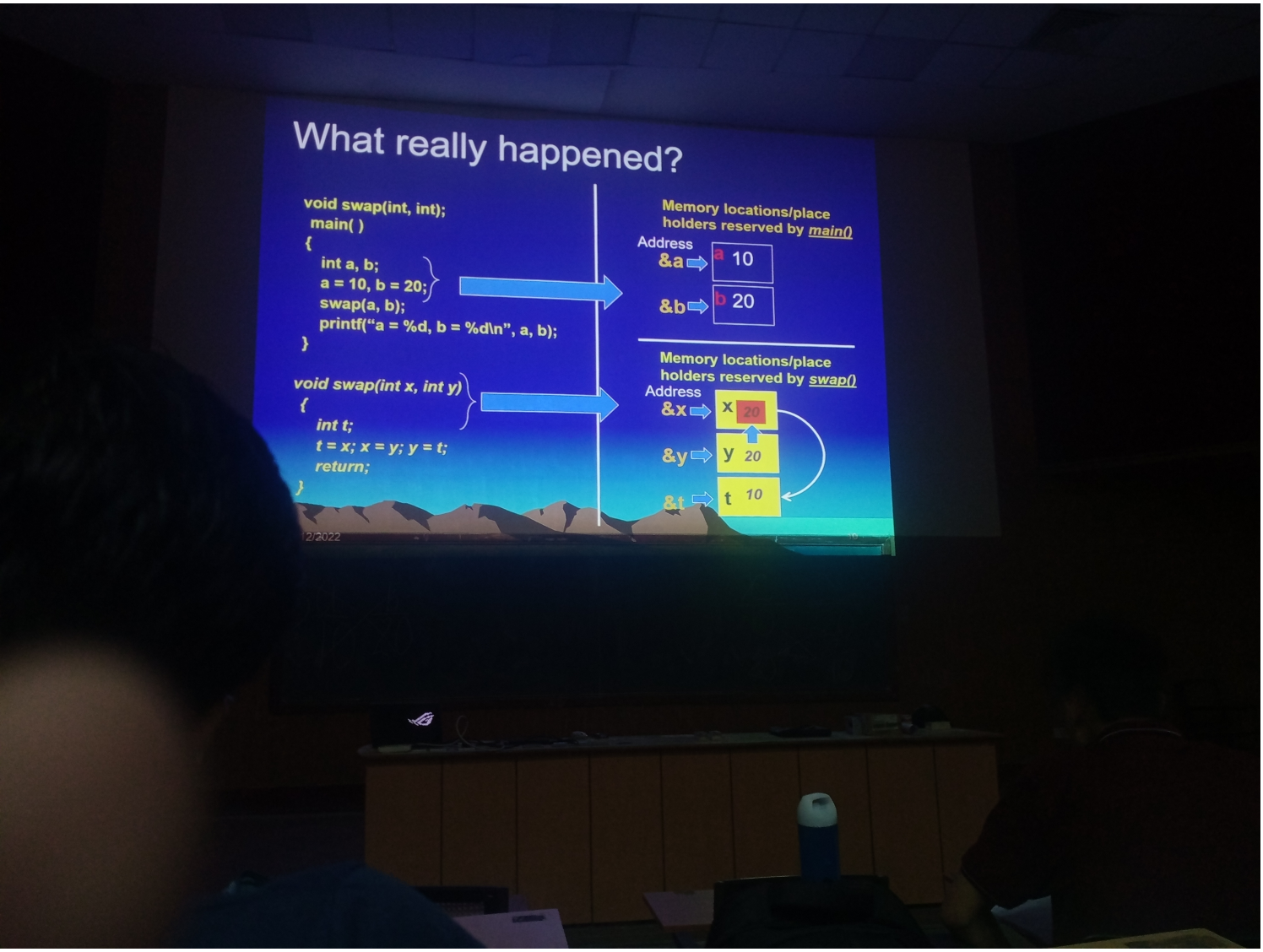

Start noting
A rendezvous with
int z = 20;
Compiler reserves a location to store an address (not
int *ptr; °
ptr = &z;
Execution causes the address of z to stored in ptr
Address of z i.e. &z
0x7ffff4e9de5a
z 20
Address of z is stored in this memory location
Address of ptr
0x7ffff4e9deAa
ptr
0x7ffff4e9de5a
Address of the memory location
where the address of an int variable can be stored

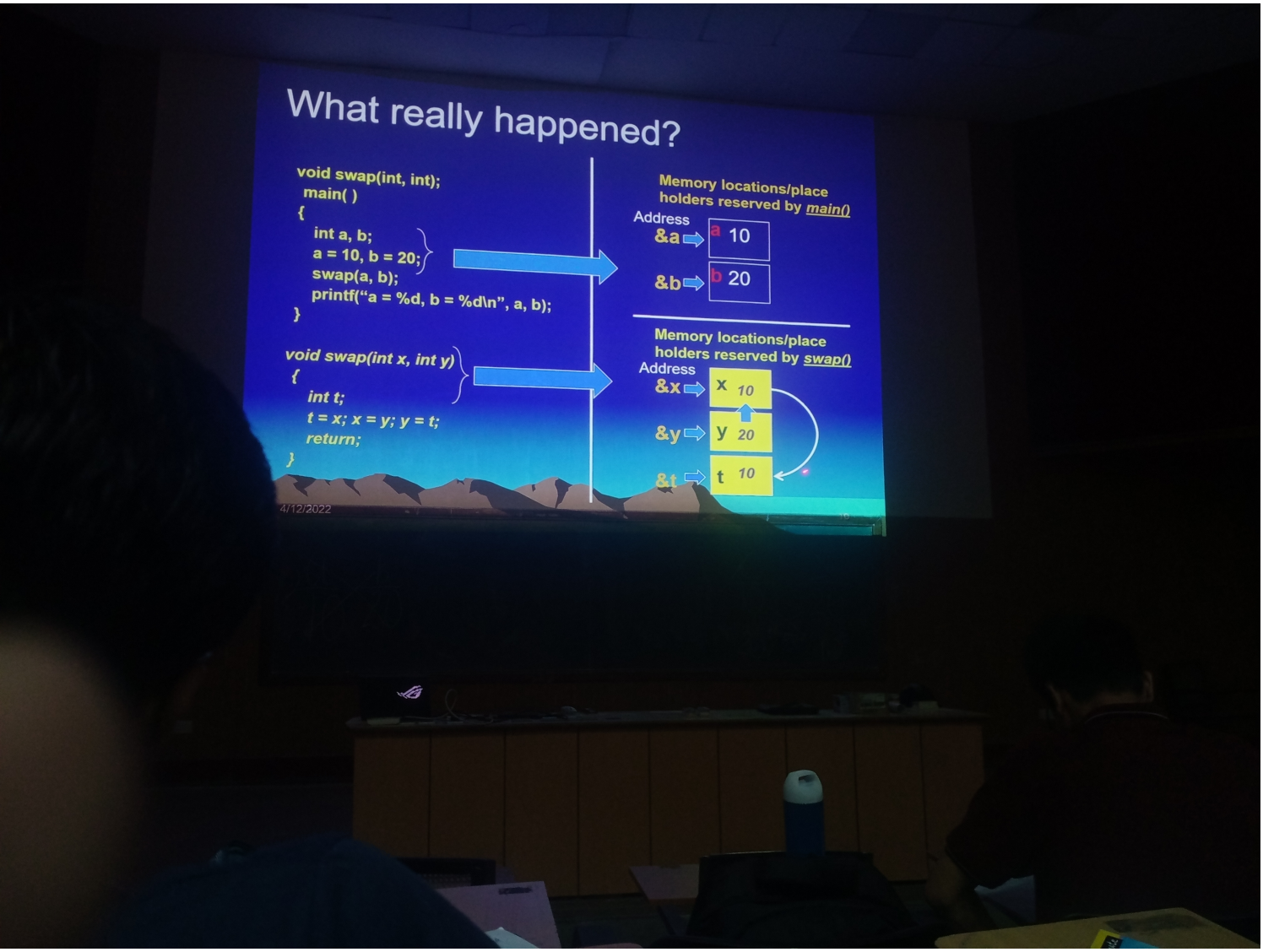

Start noting
A rendezvous with Pointers
int z = 20;
Compiler reserves a location to store an address (int)
int *ptr;
Address of z i.e. &z
0x7ffff4e9de5a
z 20
Address of z is stored in this memory location
ptr = &z;
Execution causes the address of z to be stored in ptr
Address of ptr
0x7ffff4e9deAa
ptr 0x7ffff4e9de5a
Address of the memory location where the address of an int variable can be stored

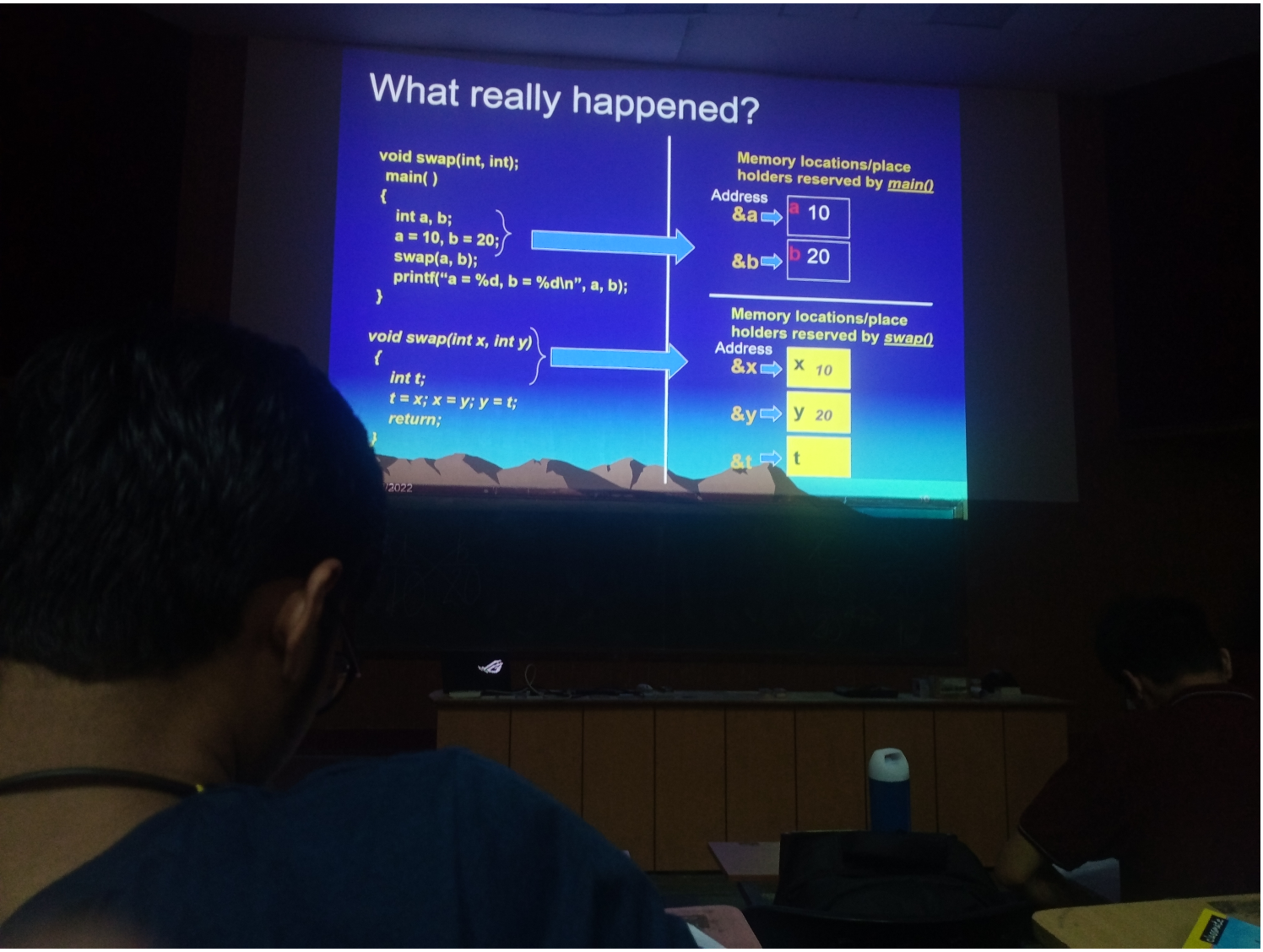

How can swap work correctly?
• Ways by which information is passed to a function.
• Passing values: This is what our swap is doing
• Passing addresses: In C, we need to program this by passing addresses.
• You now need to pass addresses of a and b viz. &a and &b
16: 19
4/12/2022

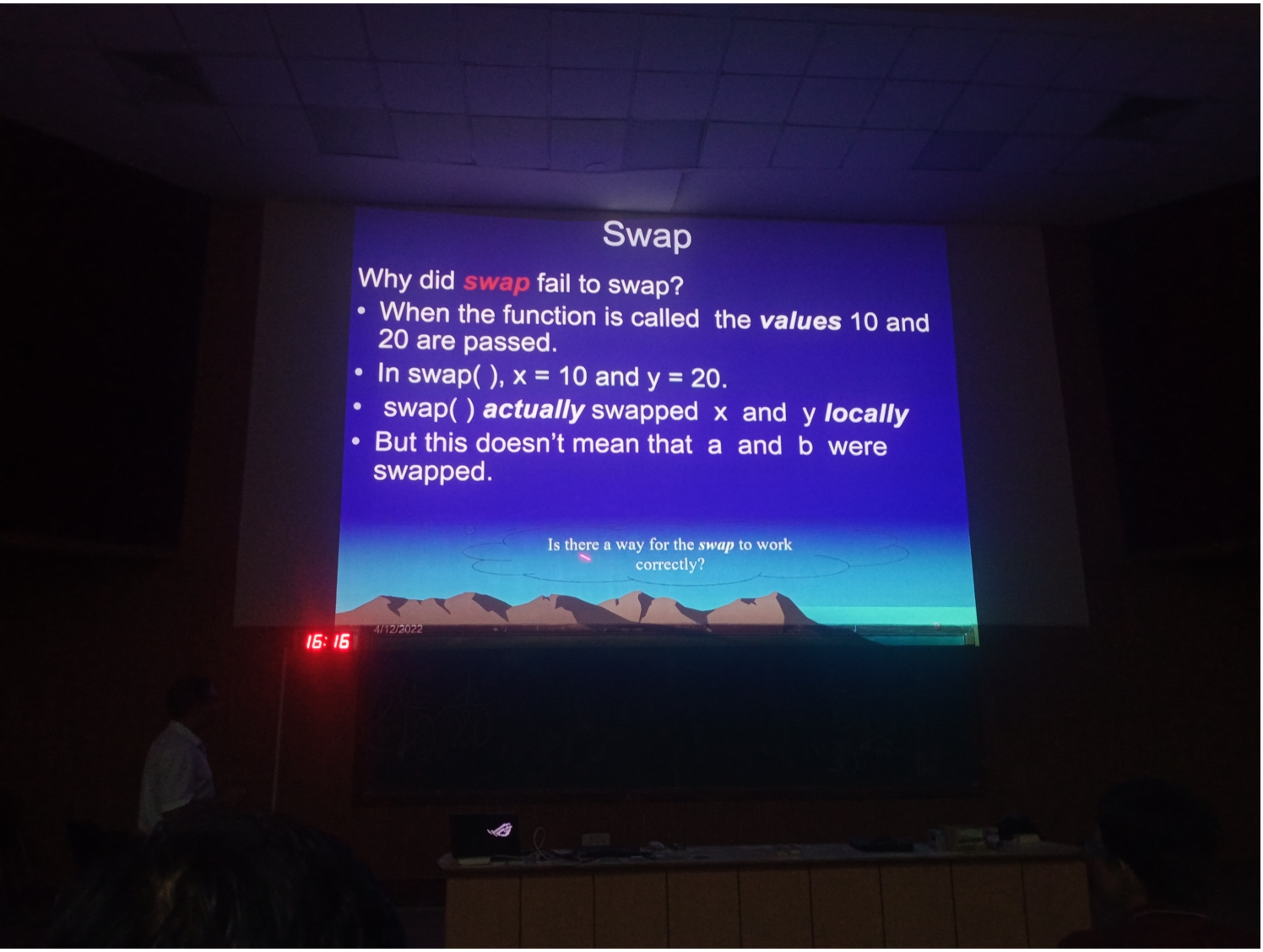

There was no change in the memory locations/place holders of main()
void swap(int, int);
main( )
{
int a, b;
a = 10, b = 20;
swap(a, b);
printf("a = %d, b = %d\n", a, b);
}
void swap(int x, int y)
{
int t;
t = x; x = y; y = t;
return;
}
Memory locations/place holders reserved by main()
Address
&a
a 10
&b
b 20
Memory locations/place holders reserved by swap()
Address
&x
x 20
&y
y 10
&t
t 10
12/2022


What really happened?
void swap(int, int);
main( )
{
int a, b;
a = 10, b = 20;
swap(a, b);
printf("a = %d, b = %d\n", a, b);
}
void swap(int x, int y)
{
int t;
t = x; x = y; y = t;
return;
}
Memory locations/place holders reserved by main()
Address
&a
a 10
&b
b 20
Memory locations/place holders reserved by swap()
Address
&x
x 20
&y
y 20
&t
t 10
4/12/2022


What really happened?
void swap(int, int);
main( )
{
int a, b;
a = 10, b = 20;
swap(a, b);
printf("a = %d, b = %d\n", a, b);
}
void swap(int x, int y)
{
int t;
t = x; x = y; y = t;
return;
}
Memory locations/place holders reserved by main()
Address
&a
a 10
&b
b 20
Memory locations/place holders reserved by swap()
Address
&x
x 20
&y
y 20
&t
t 10
2/2022


What really happened?
void swap(int, int);
main( )
{
int a, b;
a = 10, b = 20;
swap(a, b);
printf("a = %d, b = %d\n", a, b);
}
void swap(int x, int y)
{
int t;
t = x; x = y; y = t;
return;
}
Memory locations/place holders reserved by main()
Address
&a
a 10
&b
b 20
Memory locations/place holders reserved by swap()
Address
&x
x 10
&y
y 20
&t
t 10
4/12/2022


What really happened?
void swap(int, int);
main( )
{
int a, b;
a = 10, b = 20;
swap(a, b);
printf("a = %d, b = %d\n", a, b);
}
void swap(int x, int y)
{
int t;
t = x; x = y; y = t;
return;
}
Memory locations/place holders reserved by main()
Address
&a
a 10
&b
b 20
Memory locations/place holders reserved by swap()
Address
&x
x 10
&y
y 20
&t
t


Swap
Why did swap fail to swap?
• When the function is called the values 10 and 20 are passed.
• In swap( ), x = 10 and y = 20.
• swap( ) actually swapped x and y locally
• But this doesn't mean that a and b were swapped.
Is there a way for the swap to work correctly?
4/12/2022
16: 16