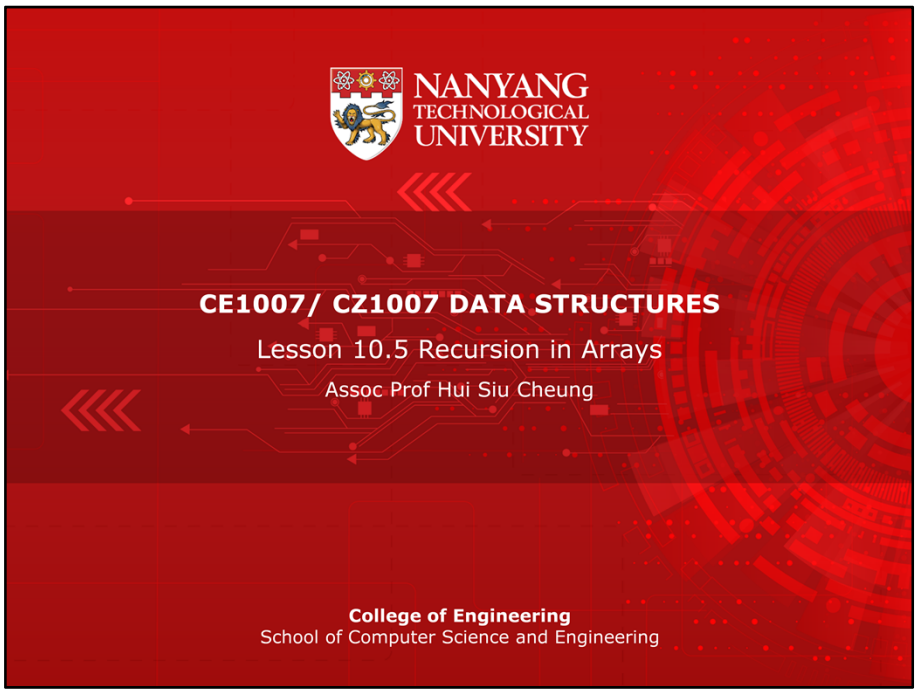
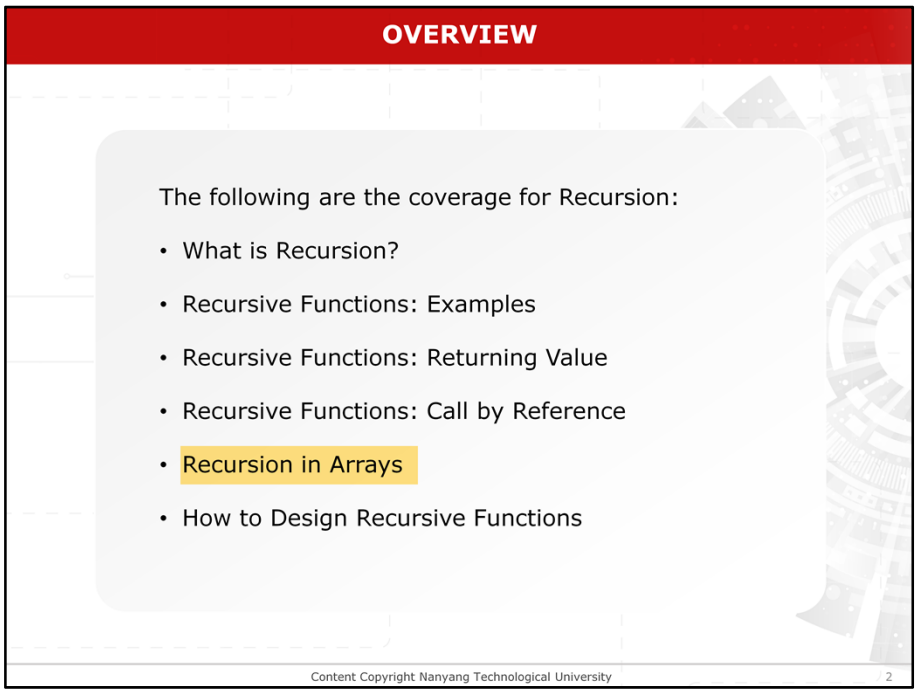
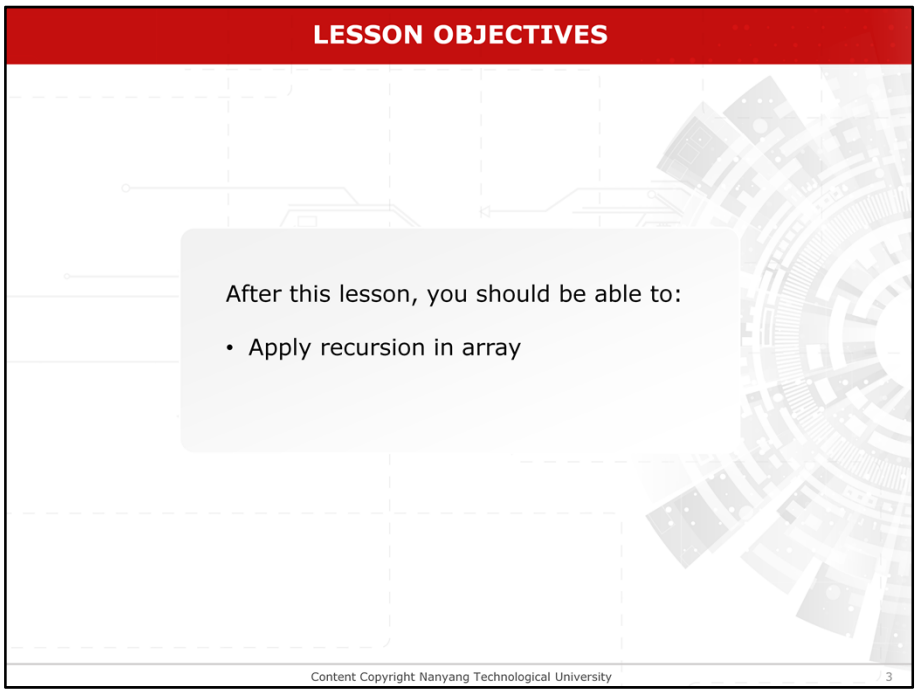
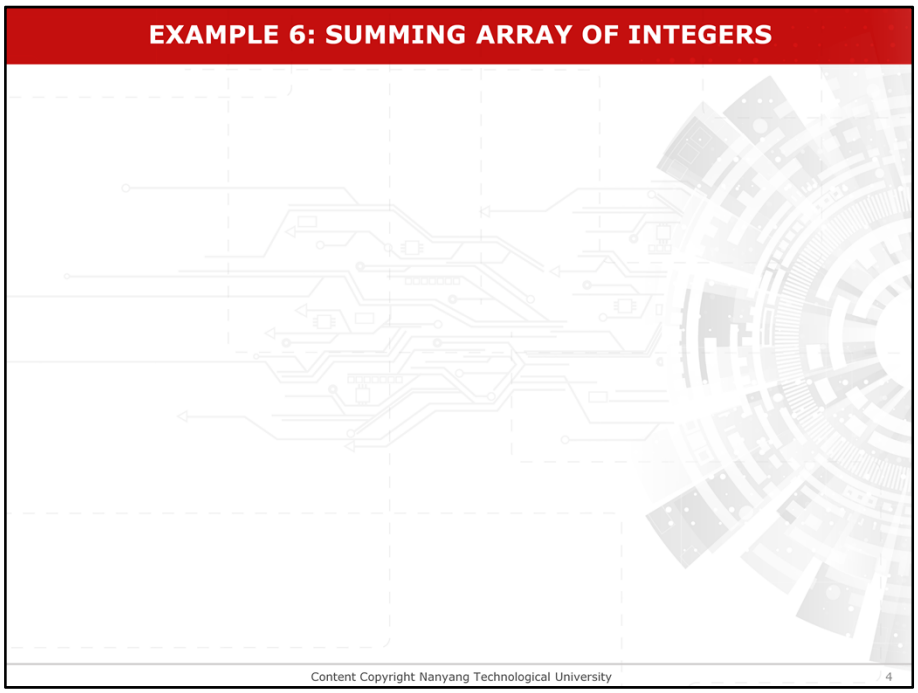

NANYANG
TECHNOLOGICAL
UNIVERSITY
CE1007/ CZ1007 DATA STRUCTURES
Lesson 10.5 Recursion in Arrays
Assoc Prof Hui Siu Cheung
College of Engineering
School of Computer Science and Engineering



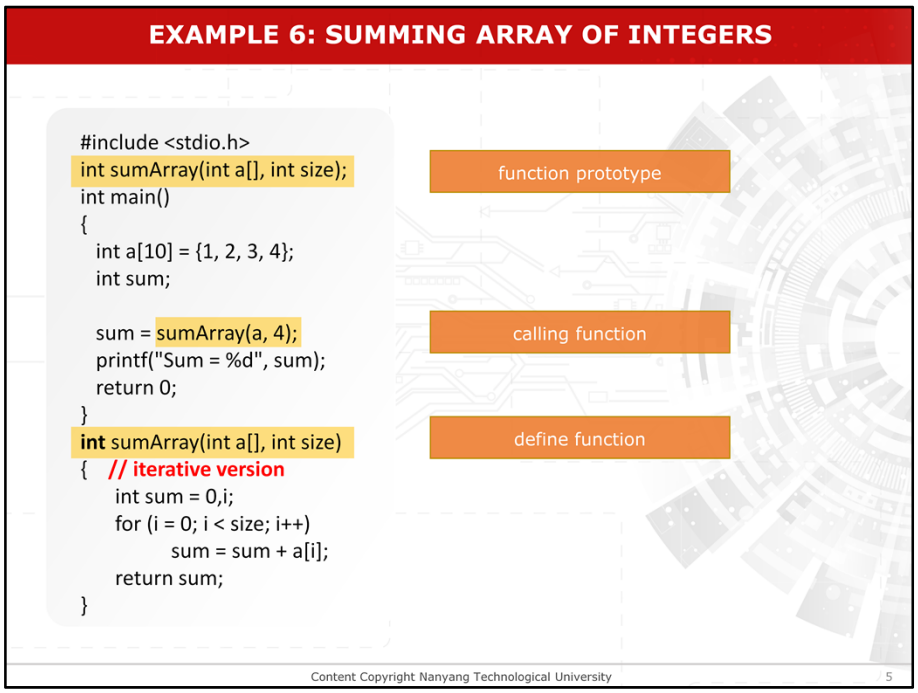
There are 6 main sections to cover for Recursion as shown. This video lesson focuses on the fifth part where we will look into recursion in arrays.



After this lesson, you should be able to apply recursion in array.



Example 6: Summing Array of Integers

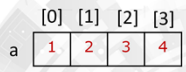


To implement a function **sum Array()** that computes the sum of all the elements in an array using non-recursive approach is quite straightforward.

A sum Array function can be written.

EXAMPLE 6: SUMMING ARRAY OF INTEGERS

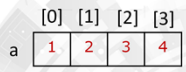
```
#include <stdio.h>
int sumArray(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;

  sum = sumArray(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int sumArray(int a[], int size)
{   // iterative version
    int sum = 0,i;
    for (i = 0; i < size; i++)
            sum = sum + a[i];
    return sum;
}
```



The array is defined as shown.

EXAMPLE 6: SUMMING ARRAY OF INTEGERS

```
#include <stdio.h>
int sumArray(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;

  sum = sumArray(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int sumArray(int a[], int size)
{   // iterative version
    int sum = 0,i;
    for (i = 0; i < size; i++)
        sum = sum + a[i];
    return sum;
}
```



Non-recursive function using for loop

A **for** loop can be used to traverse individual elements of array and add them up

EXAMPLE 6: SUMMING ARRAY OF INTEGERS

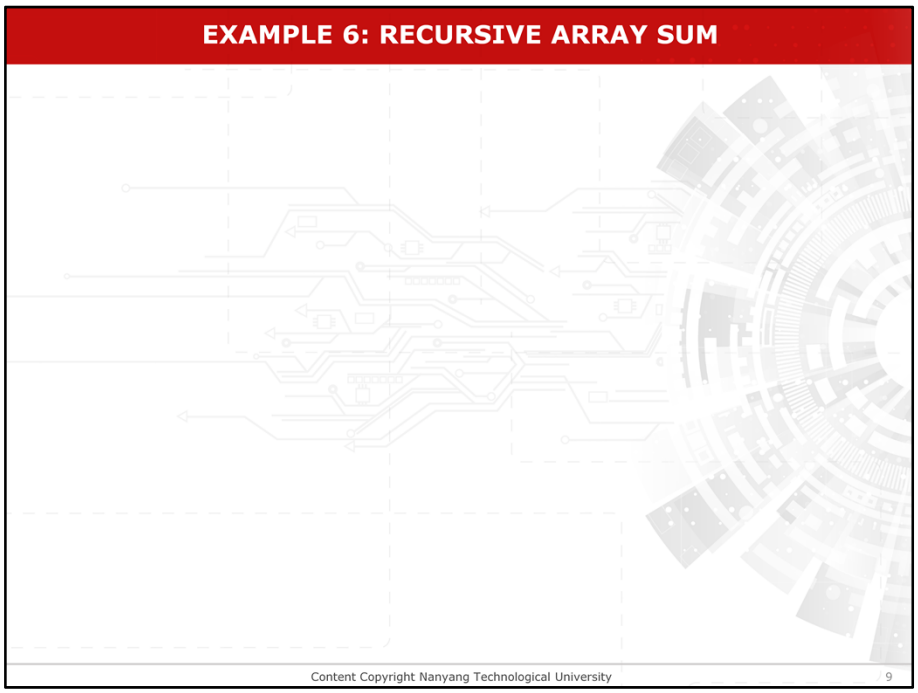
```
#include <stdio.h>
int sumArray(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;

  sum = sumArray(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int sumArray(int a[], int size)
{   // iterative version
     int sum = 0,i;
     for (i = 0; i < size; i++)
            sum = sum + a[i];
     return sum;
}
```

returns the result to the calling function

Output

Sum = 10

and returns the result to the calling function.

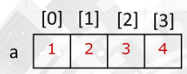


We have just seen how to use a non-recursive approach by using for loop to sum an array of integers.

We can also use a recursive approach to give the result of the summation. We will see how to apply recursion on the same example.

EXAMPLE 6: RECURSIVE ARRAY SUM

```
#include <stdio.h>
int recursiveSum(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;
  sum = recursiveSum(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int recursiveSum(int a[ ], int size)
{   // recursive version
    if (size == 1)      /* terminating condition*/
        return a[0];
    else                /* recursive condition*/
        return a[size-1] + recursiveSum(a, size-1);
}
```



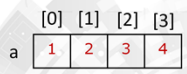
Recursive approach

This is the code using recursive approach.

To implement the function that recursively computes the sum of elements in arrays is more challenging.

EXAMPLE 6: RECURSIVE ARRAY SUM

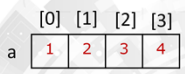
```
#include <stdio.h>
int recursiveSum(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;
  sum = recursiveSum(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int recursiveSum(int a[ ], int size)
{   // recursive version
    if (size == 1)      /* terminating condition*/
        return a[0];
    else                /* recursive condition*/
        return a[size-1] + recursiveSum(a, size-1);
}
```



The function **recursive Sum()** has two parameters: **a** is an array and **size** is the size of the array to be processed.

EXAMPLE 6: RECURSIVE ARRAY SUM

```
#include <stdio.h>
int recursiveSum(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;
  sum = recursiveSum(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int recursiveSum(int a[ ], int size)
{   // recursive version
    if (size == 1)      /* terminating condition*/
        return a[0];
    else                /* recursive condition*/
        return a[size-1] + recursiveSum(a, size-1);
}
```

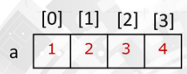


recursive function

In the function **recursive Sum()**, it computes the sum of all the elements in the array by recursively processes the array and reduces the size of the array by one element each time until the terminating condition is reached. In which case, the array will contain only one element.

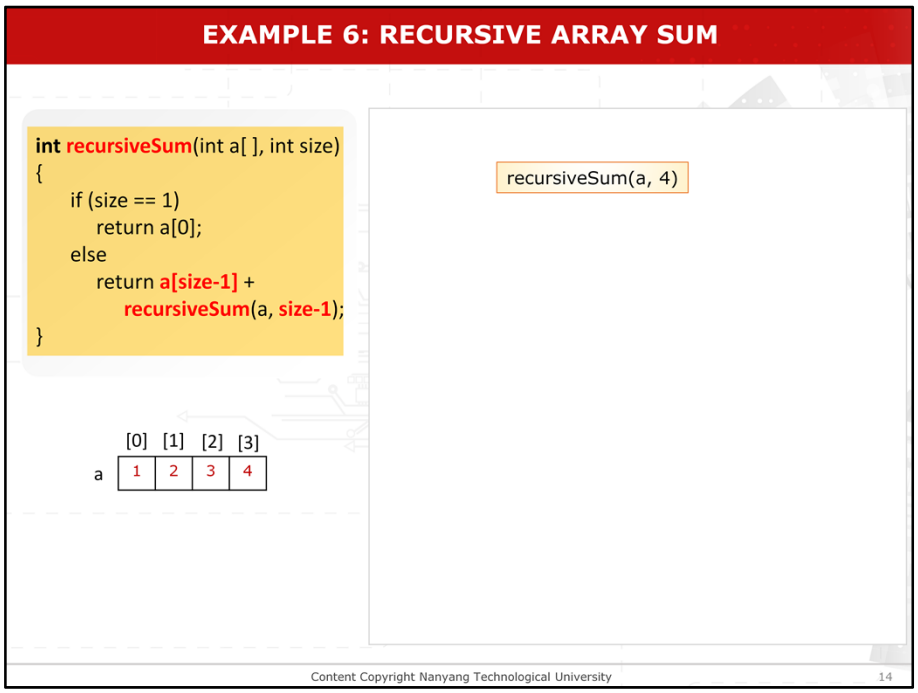
EXAMPLE 6: RECURSIVE ARRAY SUM

```
#include <stdio.h>
int recursiveSum(int a[], int size);
int main()
{
  int a[10] = {1, 2, 3, 4};
  int sum;
  sum = recursiveSum(a, 4);
  printf("Sum = %d", sum);
  return 0;
}
int recursiveSum(int a[ ], int size)
{   // recursive version
    if (size == 1)      /* terminating condition*/
        return a[0];
    else                /* recursive condition*/
        return a[size-1] + recursiveSum(a, size-1);
}
```

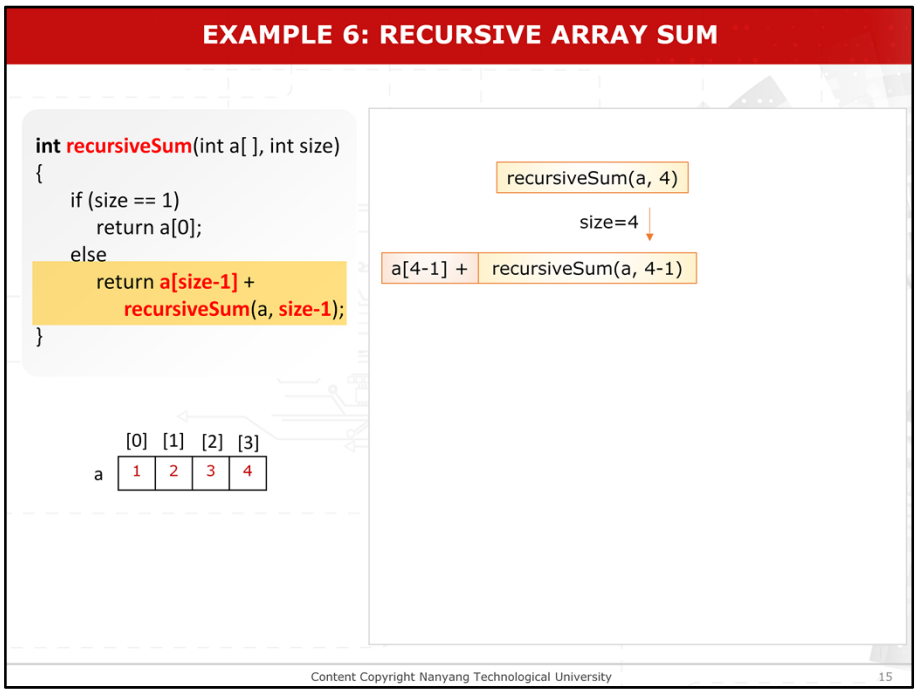


recursive function

We will look into detail the tracing of the recursive function when **recursive Sum (a, 4)** is called from the main() function.



The highlighted code is the recursive function which we will look in detail how **recursive Sum (a, 4)** will be executed.

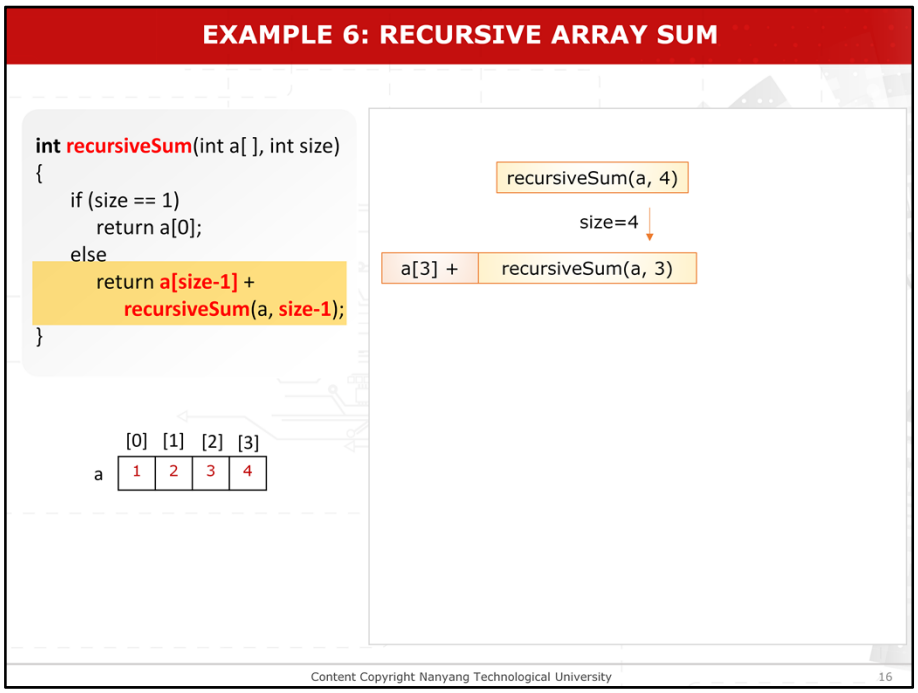


Size is equal to 4 so the code under the recursive condition is executed as **size** is not equal to 1.

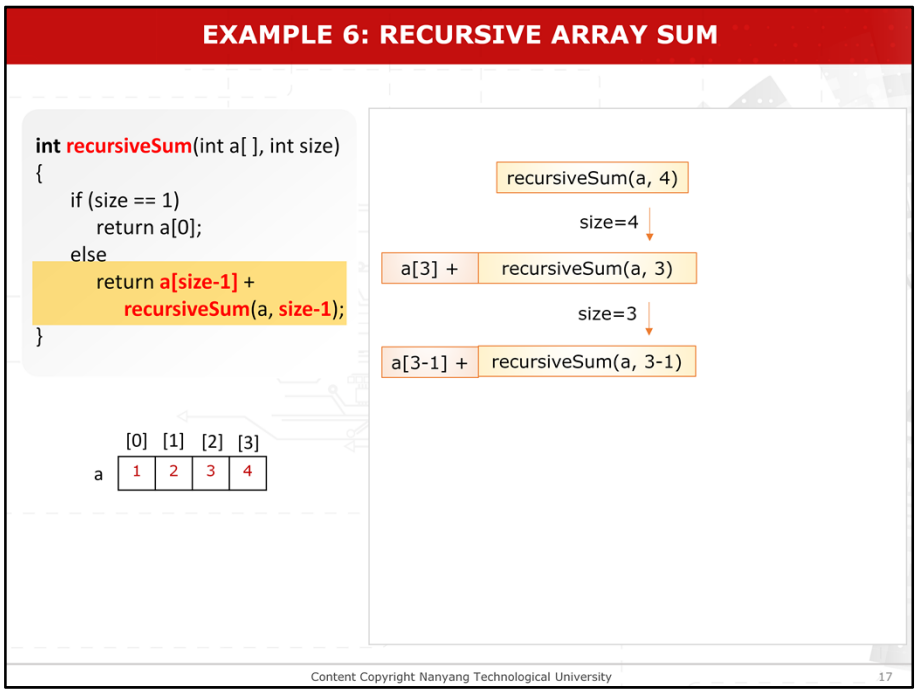
The statement

return array a [size-1] + recursive Sum(a, size-1);

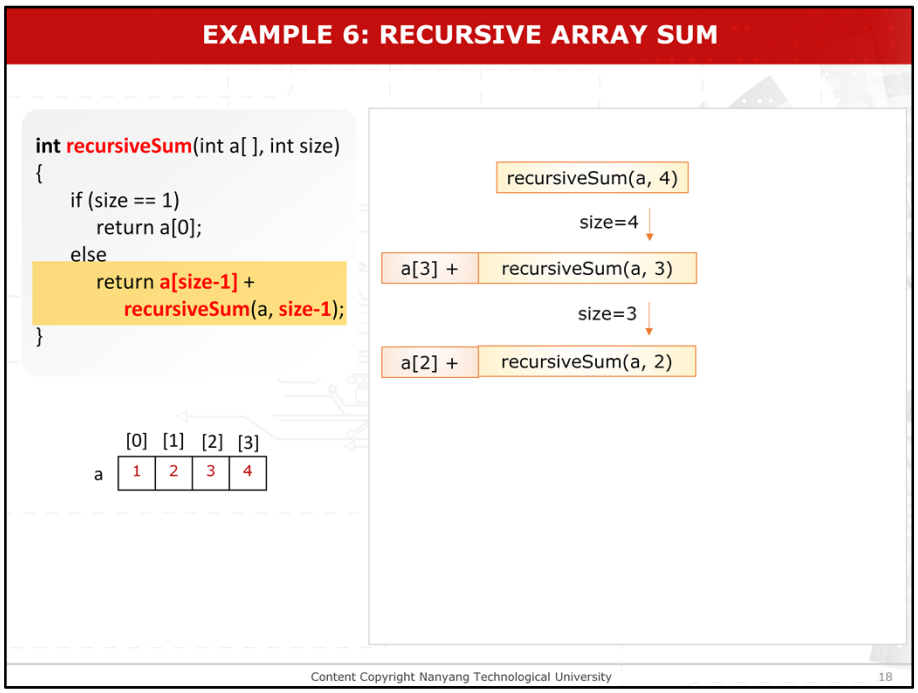
will be executed



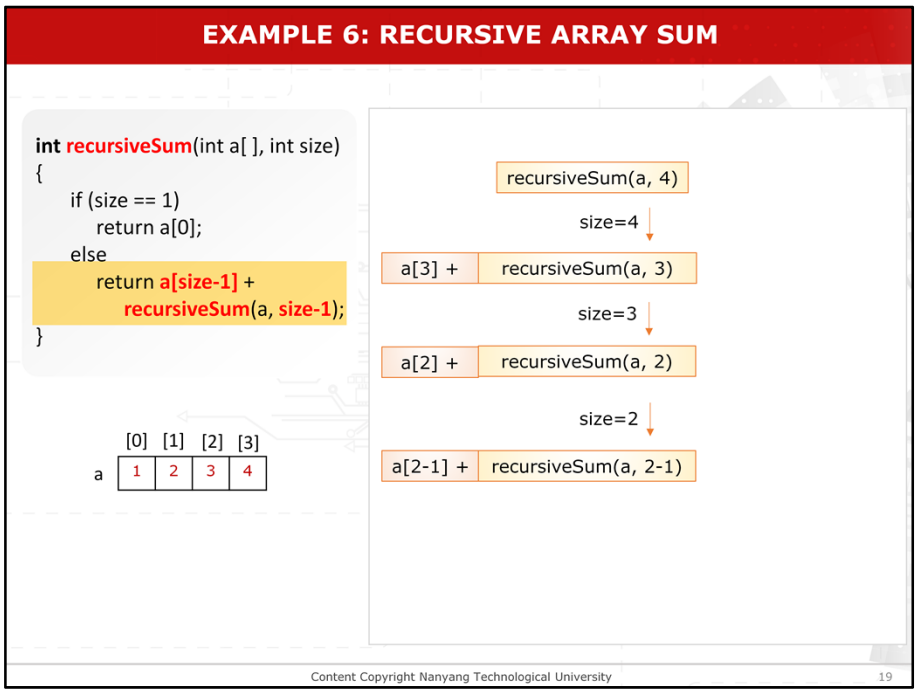
and call the function **recursiveSum(a, 3)**



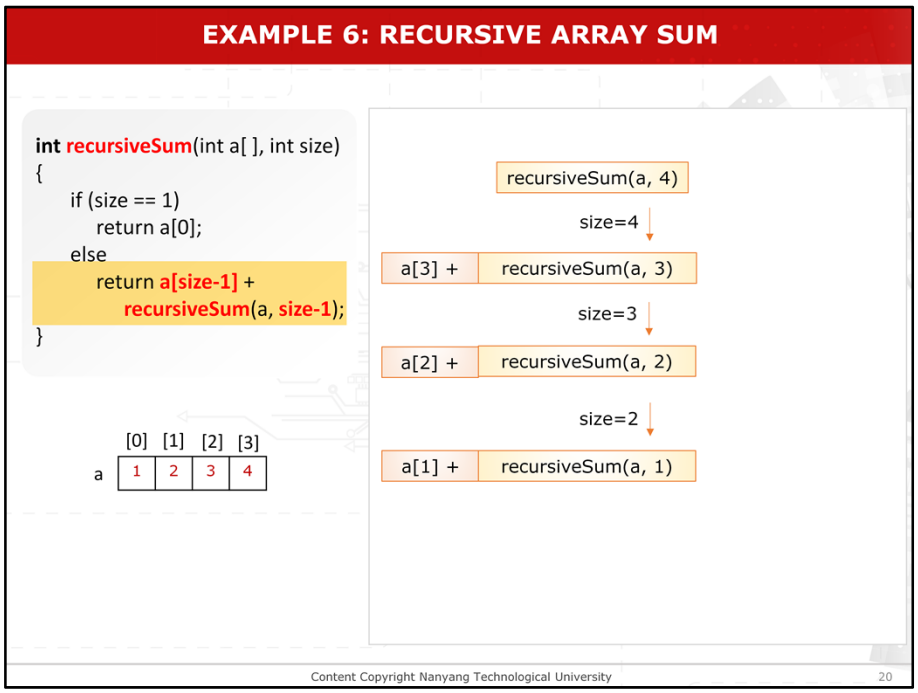
with **size** reduced to 3.



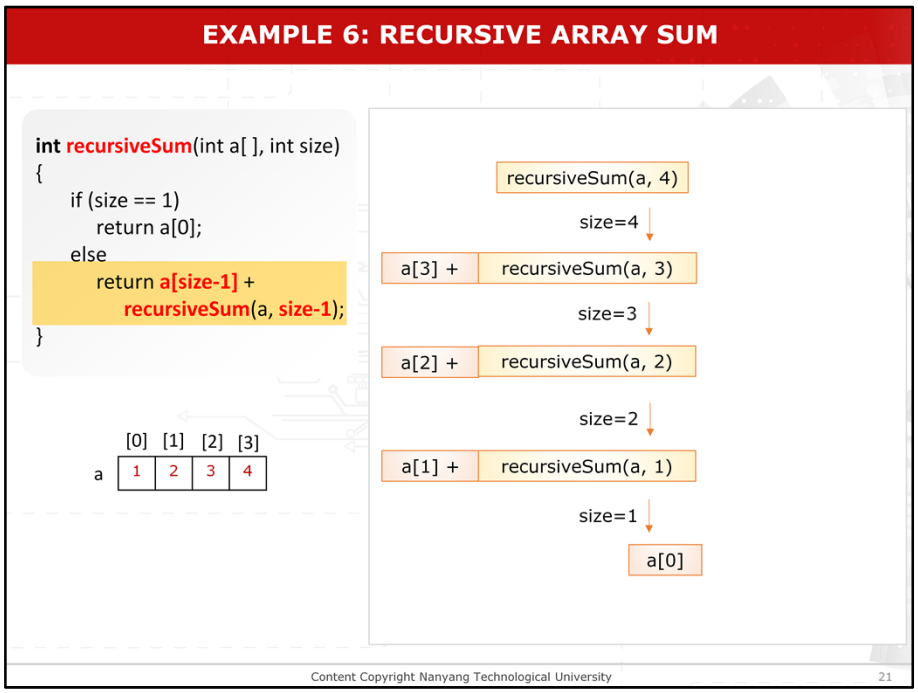
This will in turn execute the function call **recursiveSum(a, 2)**.


EXAMPLE 6: RECURSIVE ARRAY SUM
int recursiveSum(int a[ ], int size)
{
if (size == 1)
return a[0];
else
return a[size-1] +
recursiveSum(a, size-1);
}
[0] [1] [2] [3]
a 1 2 3 4
recursiveSum(a, 4)
size=4
a[3] + recursiveSum(a, 3)
size=3
a[2] + recursiveSum(a, 2)
size=2
a[2-1] + recursiveSum(a, 2-1)
Content Copyright Nanyang Technological University
19

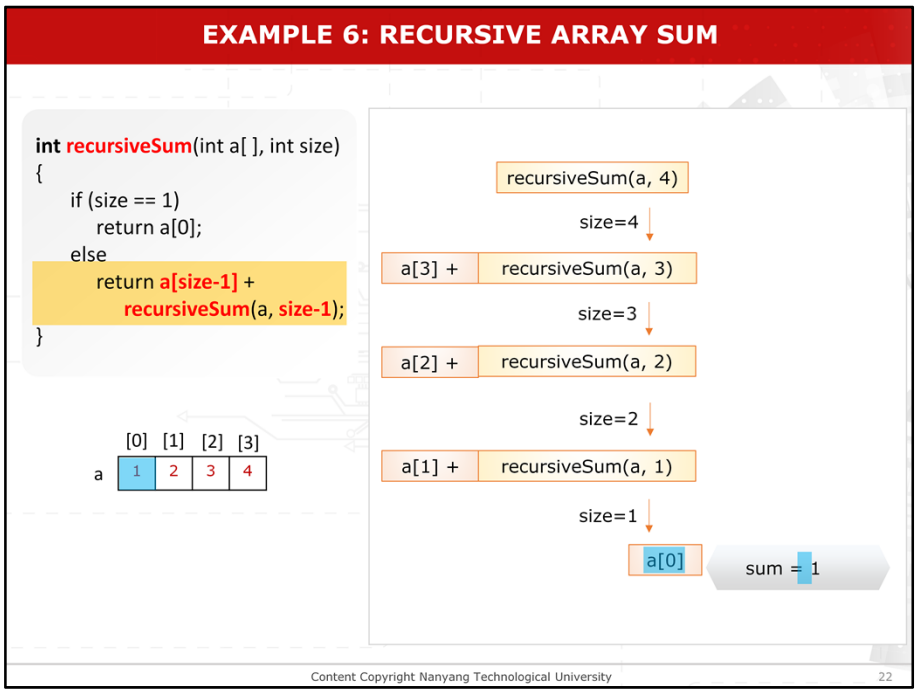
[no audio]



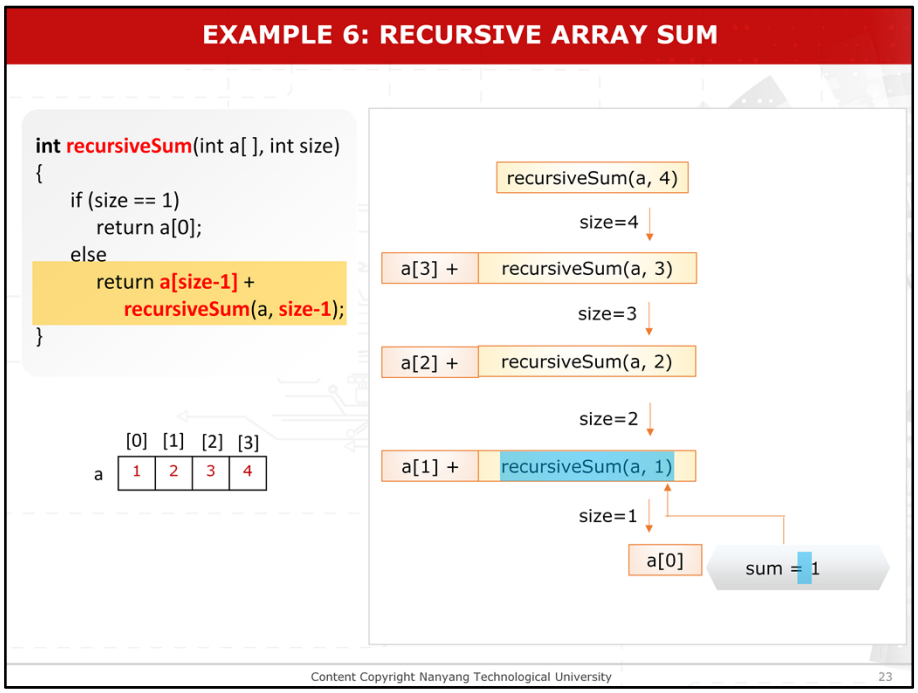
When the function **recursiveSum(a, 1)** is executed,



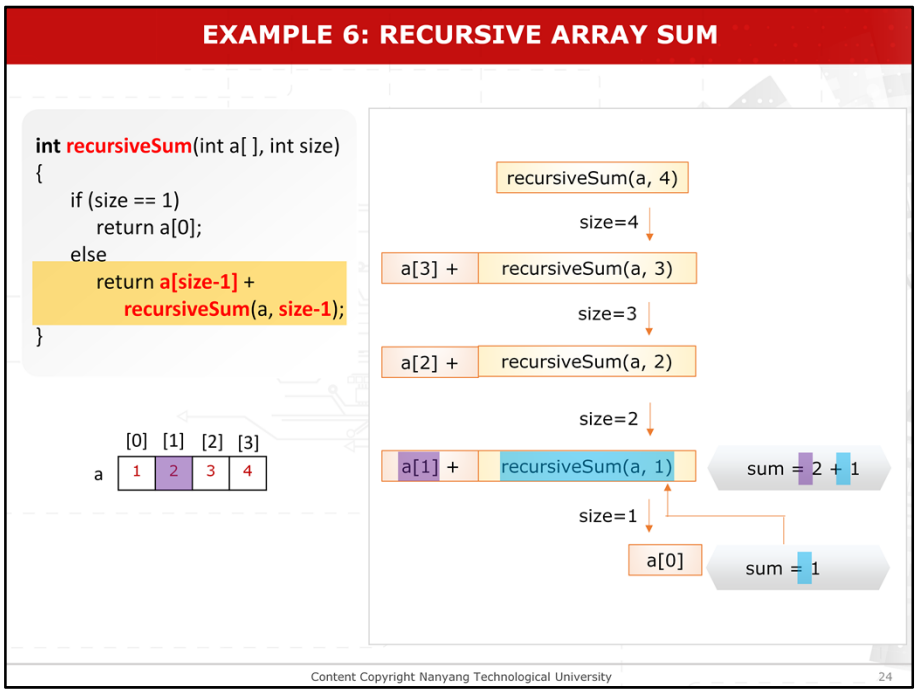
the code under the terminating condition will be executed.



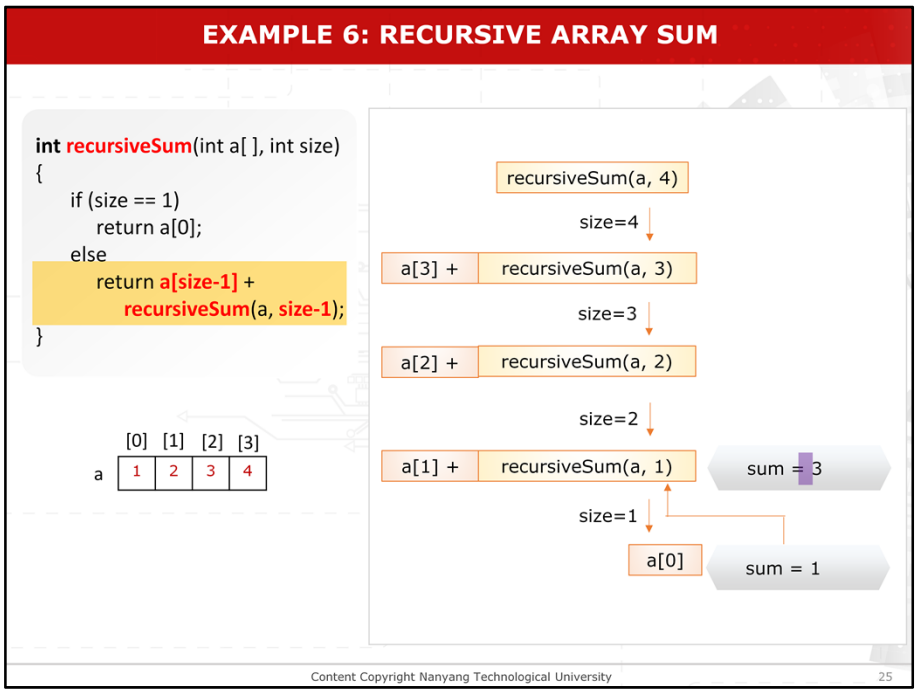
The value of **a[0]** (that is 1)



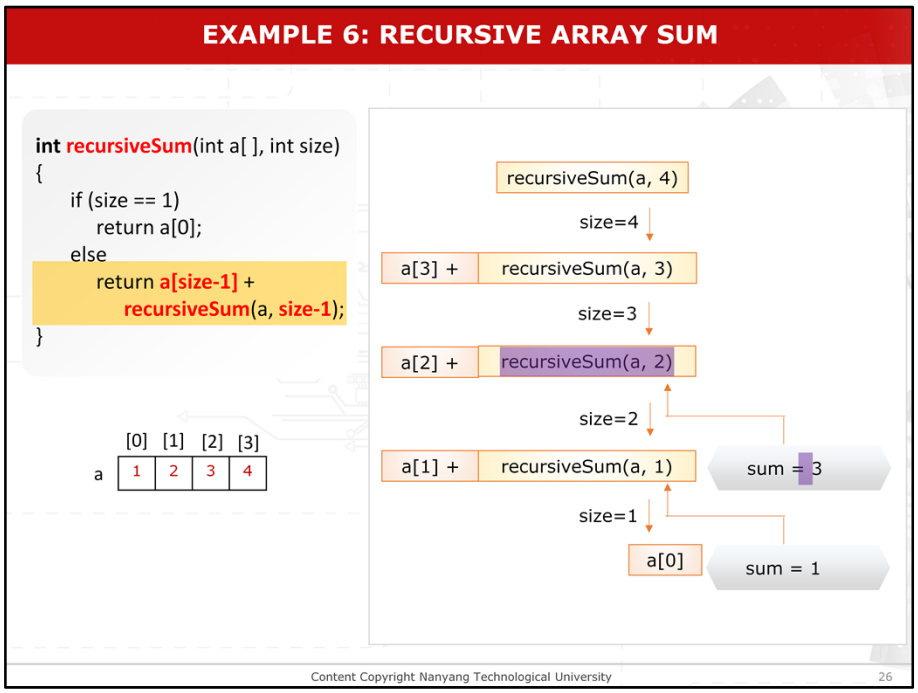
will be returned to the calling function.



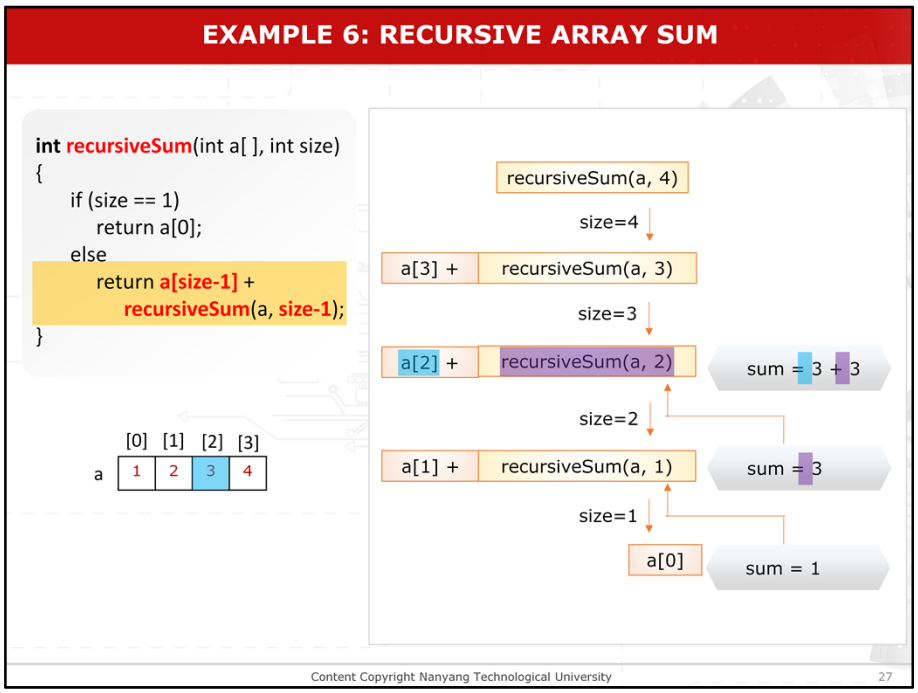
The previous calling function will receive the value 1 and add the returned value to **a[1]**


EXAMPLE 6: RECURSIVE ARRAY SUM
int recursiveSum(int a[ ], int size)
{
if (size == 1)
return a[0];
else
return a[size-1] +
recursiveSum(a, size-1);
}
[0] [1] [2] [3]
a 1 2 3 4
recursiveSum(a, 4)
size=4
a[3] + recursiveSum(a, 3)
size=3
a[2] + recursiveSum(a, 2)
size=2
a[1] + recursiveSum(a, 1)
sum = 3
size=1
a[0]
sum = 1
Content Copyright Nanyang Technological University
25

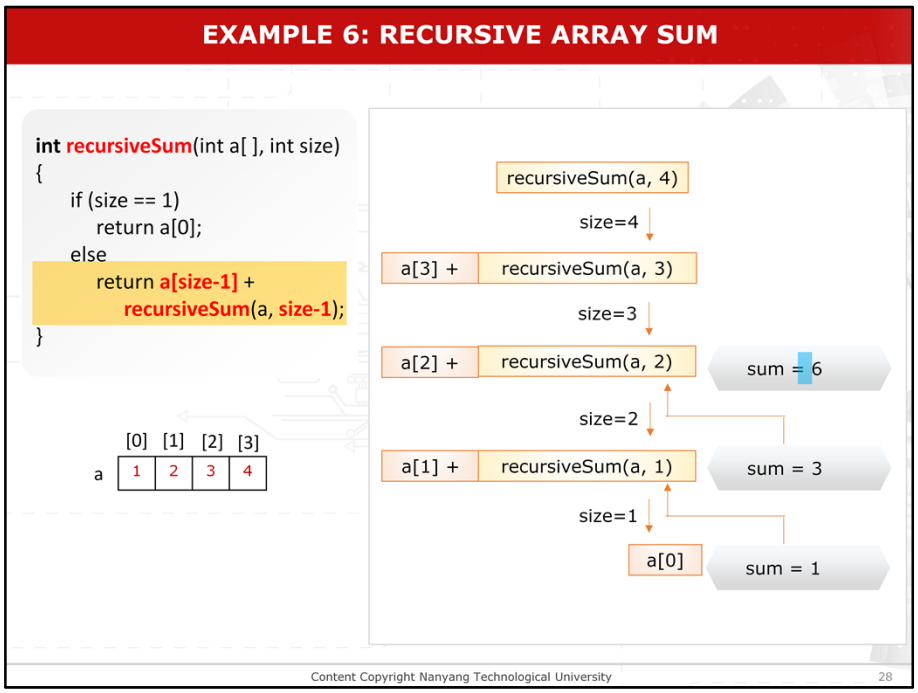
[no audio]



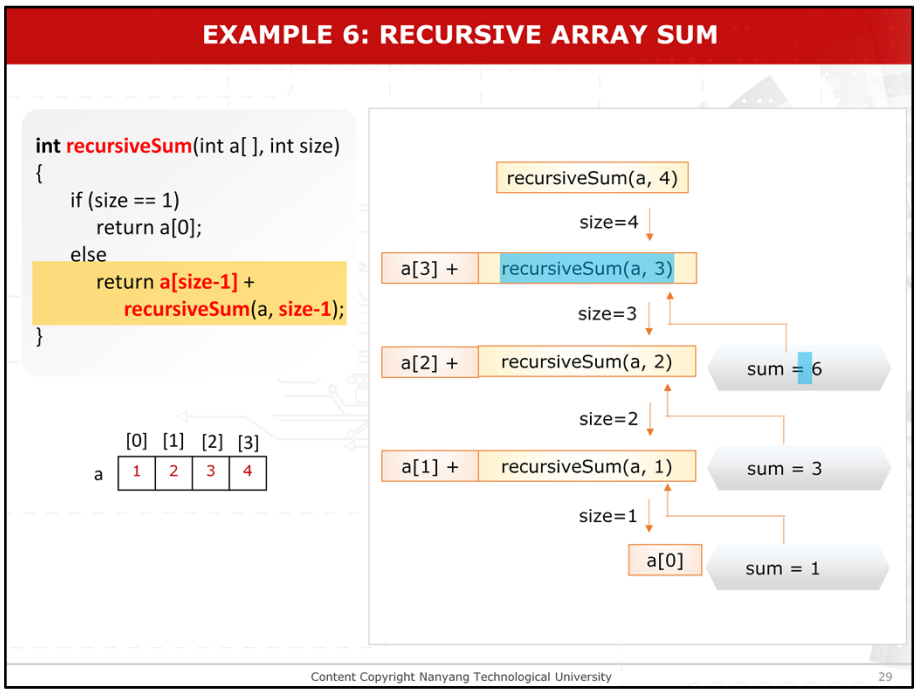
The sum of 3 is then returned to the previous calling function.



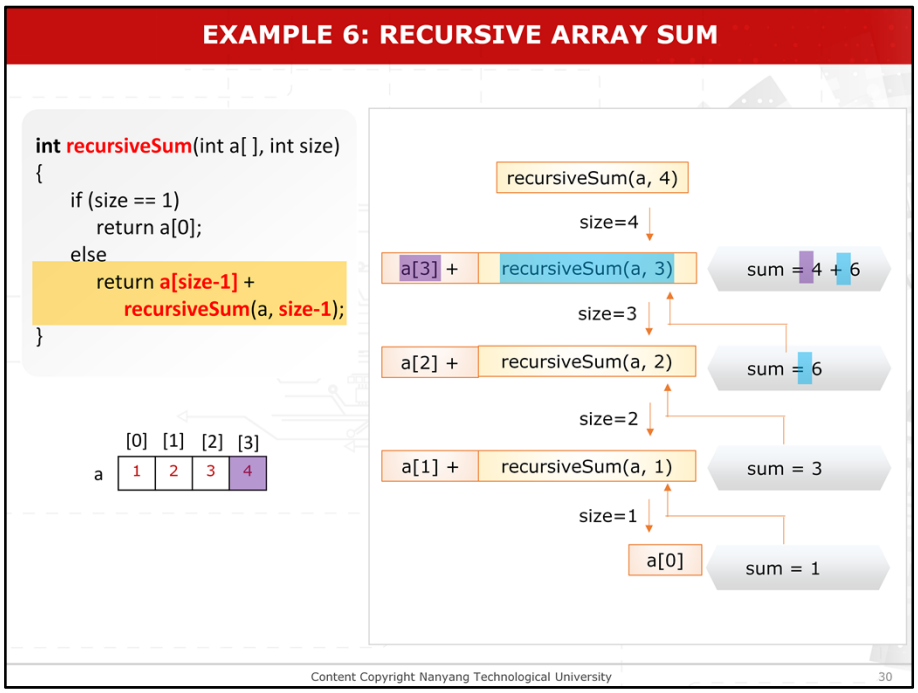
The function will add the returned value of 3 to **a[2]**.


EXAMPLE 6: RECURSIVE ARRAY SUM
int recursiveSum(int a[ ], int size)
{
if (size == 1)
return a[0];
else
return a[size-1] +
recursiveSum(a, size-1);
}
[0] [1] [2] [3]
a 1 2 3 4
recursiveSum(a, 4)
size=4
a[3] + recursiveSum(a, 3)
size=3
a[2] + recursiveSum(a, 2)
sum = 6
size=2
a[1] + recursiveSum(a, 1)
sum = 3
size=1
a[0]
sum = 1
Content Copyright Nanyang Technological University
28

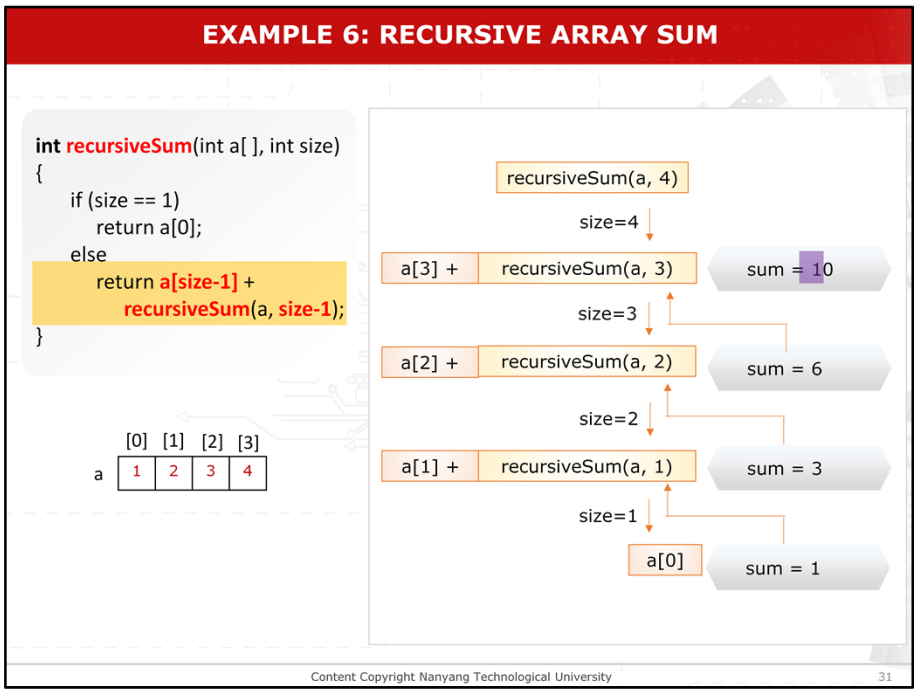
[no audio]



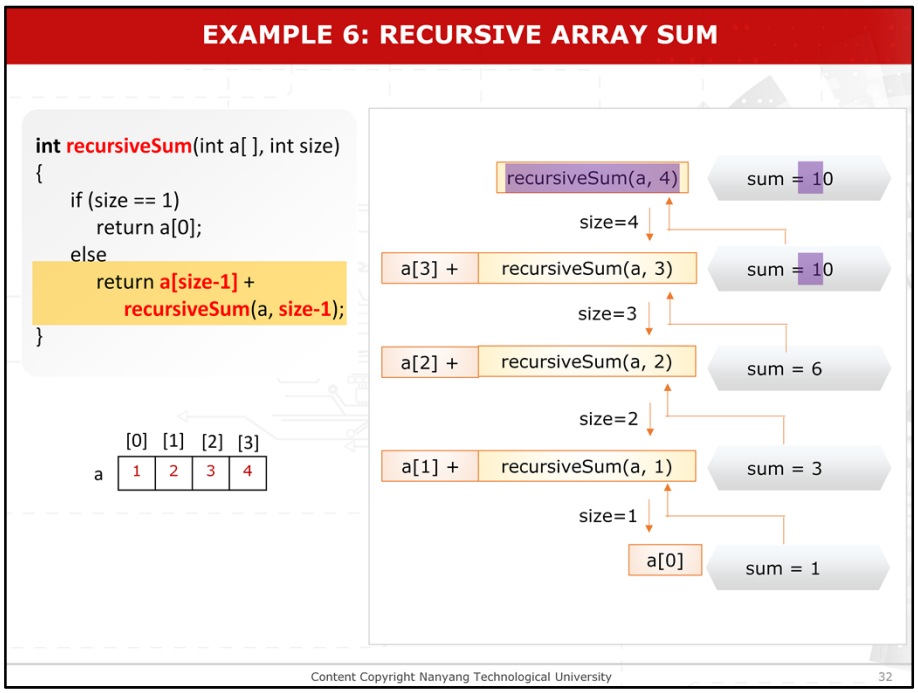
the sum of 6 is then returned to the previous calling function



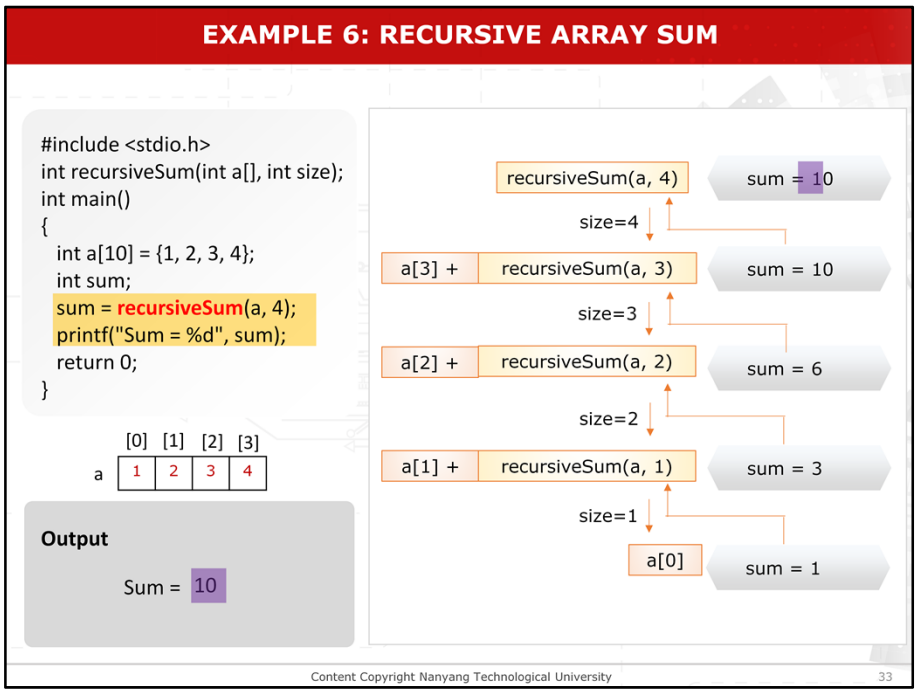
The function will add the returned value of 6 to **a[3]**


EXAMPLE 6: RECURSIVE ARRAY SUM
int recursiveSum(int a[ ], int size)
{
if (size == 1)
return a[0];
else
return a[size-1] +
recursiveSum(a, size-1);
}
[0] [1] [2] [3]
a 1 2 3 4
recursiveSum(a, 4)
size=4
a[3] + recursiveSum(a, 3)
sum = 10
size=3
a[2] + recursiveSum(a, 2)
sum = 6
size=2
a[1] + recursiveSum(a, 1)
sum = 3
size=1
a[0]
sum = 1
Content Copyright Nanyang Technological University
31

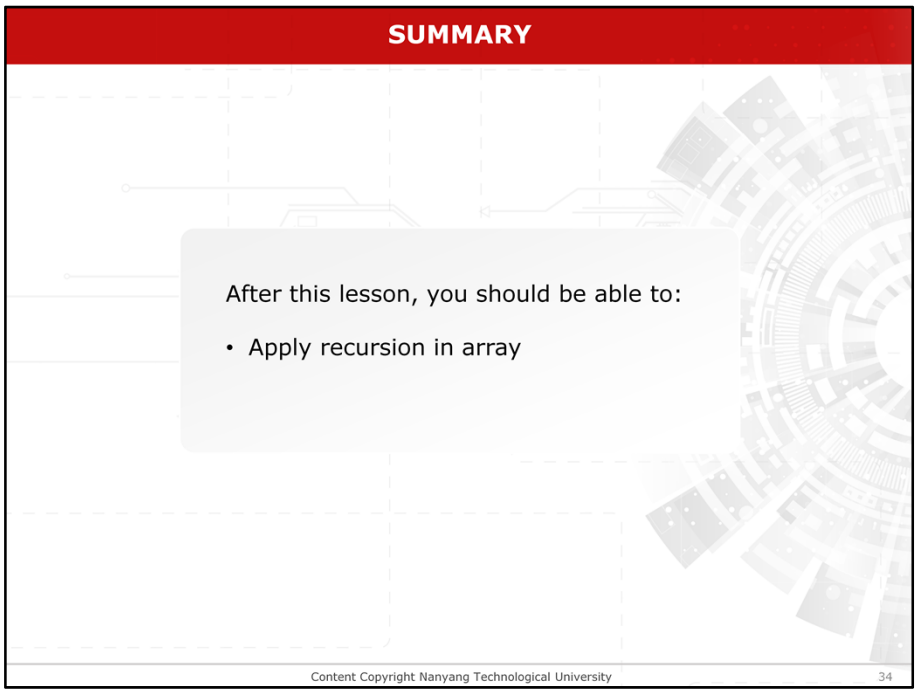
[no audio]



The sum of 10 will be returned to the calling function **main()**



In the **main()** function, the returned value of 10 will then be printed to the screen.



In summary, after viewing this video lesson, you should be able to apply recursion in array.