CSC215

Math and Computer Science

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Recursion

- Related to mathematical induction
  - Definition of the nth value in terms of the first n-1 values
  - A stopping condition

- In C++, it is a function that calls itself
  - Has what is known as a base case for stopping the function calls

Ask Yourself the Following

1. How to define the problem in terms of a smaller problem of the same type?
2. How does each recursive call diminish the size of the problem?
3. What instance of the problem can serve as the base case?
4. As the problem diminishes, will it reach the base case?

The example here are best done iteratively.

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Factorial

- N! = N * (n-1) * (n-2) * (n-3) *…….
- How to define in terms of the first n-1 terms
- N! = N * (n-1)!

- What will serve as our base case?
  - 0! = 1
  - Could use 1! But the smallest possible factorial is defined as 0! = 1

The Factorial Criteria

- N = 0 answer is 1
- N > 0 answer is N * (N-1)!

Drawing out N!

- Assume n = 6
  - 6! = 6 * 5!
  - 5! = 5 * 4!
  - 4! = 4 * 3!
  - 3! = 3 * 2!
  - 2! = 2 * 1!
  - 1! = 1 * 0!
  - 0! = 1 ← Good, I know the answer

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Substitute Going Back Up

- Answer to n = 6, 6! = 720
  - 6! = 6 * 5! = 6 * 120 = 720
  - 5! = 5 * 4! = 5 * 24 = 120
  - 4! = 4 * 3! = 4 * 6 = 24
  - 3! = 3 * 2! = 3 * 2 = 6
  - 2! = 2 * 1! = 2 * 1 = 2
  - 1! = 1 * 0! = 1 * 1 = 1
  - 0! = 1

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Writing the function

```
unsigned int factorial( unsigned int n )
{
    // write the base case first
    if( n == 0 )
        return 1;



}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Writing the function

```
unsigned int factorial( unsigned int n )
{   int ans;
    // write the base case first
    if( n == 0 )
        return 1;

    // compute the answer
    ans = n * factorial( n - 1 );
    return ans;

}
```

Stepping Through the Function

```
int main()
{
    int result;
    int n = 4;
    result = factorial ( n );
    cout << "4! = " << result << endl;
    return 0;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main
? = F(4)	result
4	n

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)
? = F(4)	Result	
4	n	
4		n
? = N *F(3)		ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)
? = F(4)	Result		
4	n		
4		n	
? = n * F(3)		ans	
3			n
? = n * F(2)			ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2)
? = F(4)	Result			
4	n			
4		n		
? = n * F(3)		ans		
3			n	
? = n * F(2)			ans	
2				n
? = n * F(1)				ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2)	F(1)
? = F(4)	Result				
4	n				
4		n			
? = n * F(3)		ans			
3			n		
? = n * F(2)			ans		
2				n	
? = n * F(1)				ans	
1					n
? = n * F(0)					ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2)	F(1)	F(0)
? = F(4)	Result					
4	n					
4		n				
? = n * F(3)		ans				
3			n			
? = n * F(2)			ans			
2				n		
? = n * F(1)				ans		
1					n	
? = n * F(0)					ans	
0						n
						ans

Base Case Reached

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2)	F(1)	F(0) = 1
? = F(4)	Result					
4	n					
4		n				
? = n * F(3)		ans				
3			n			
? = n * F(2)			ans			
2				n		
? = n * F(1)				ans		
1					n	
n * F(0) = 1					ans	

F(0) = 1

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2)	F(1) = 1
? = F(4)	Result				
4	n				
4		n			
? = n * F(3)		ans			
3			n		
? = n * F(2)			ans		
2				n	
n * F(1) = 2				ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3)	F(2) = 2
? = F(4)	Result			
4	n			
4		n		
? = n * F(3)		ans		
3			n	
n * F(2) = 6			ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4)	F(3) = 6
? = F(4)	Result		
4	n		
4		n	
n * F(3) = 24		ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, f = factorial)

Memory	main	F(4) = 24
= F(4) = 24	Result	
4	n	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

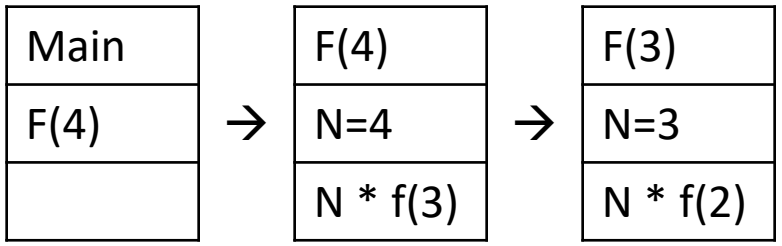
Box Method

Main
F(4)

→

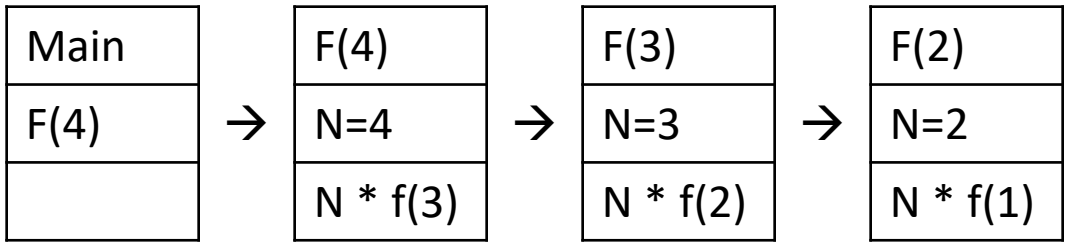
F(4)
N=4
N * f(3)

Box Method

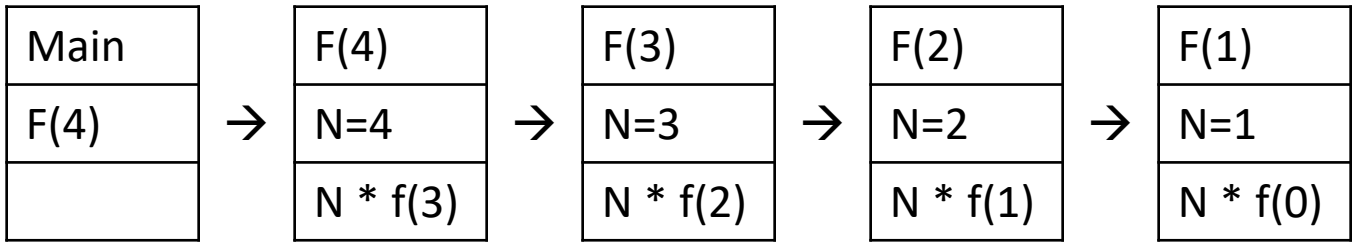


SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

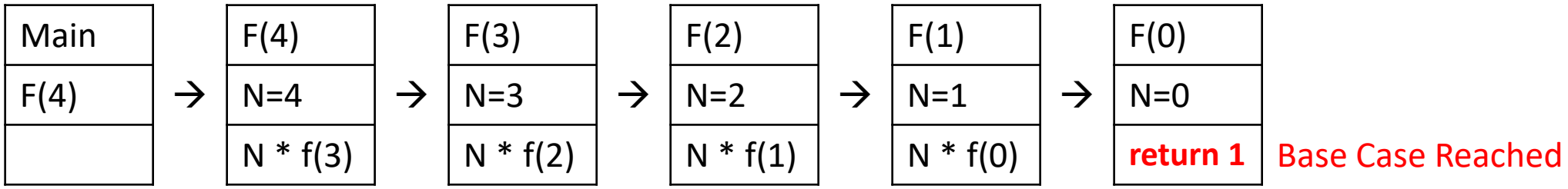
Box Method



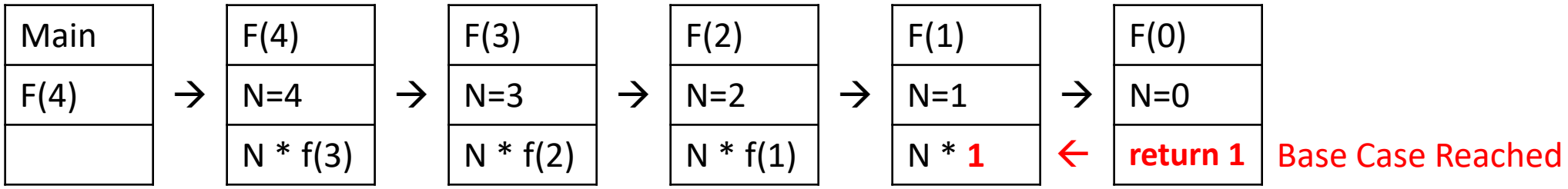
Box Method



Box Method

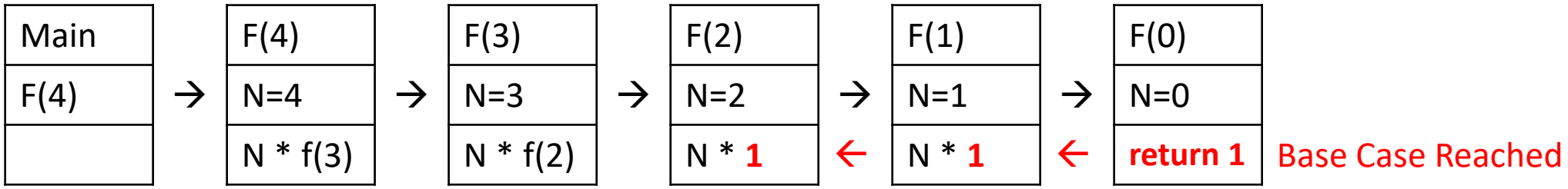

Main
F(4)
F(4)
N=4
N * f(3)
F(3)
N=3
N * f(2)
F(2)
N=2
N * f(1)
F(1)
N=1
N * f(0)
F(0)
N=0
return 1
Base Case Reached

Box Method



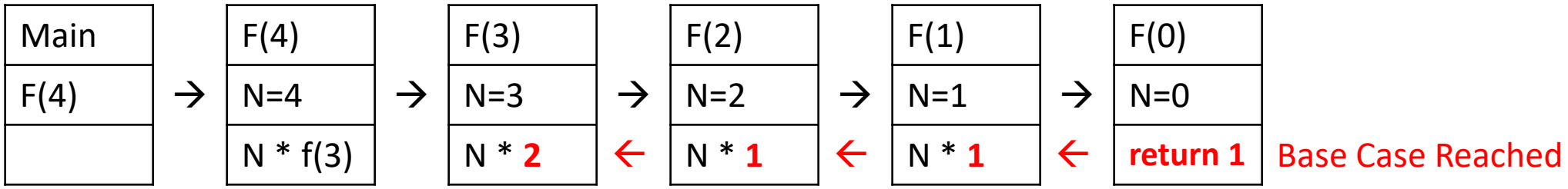
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method

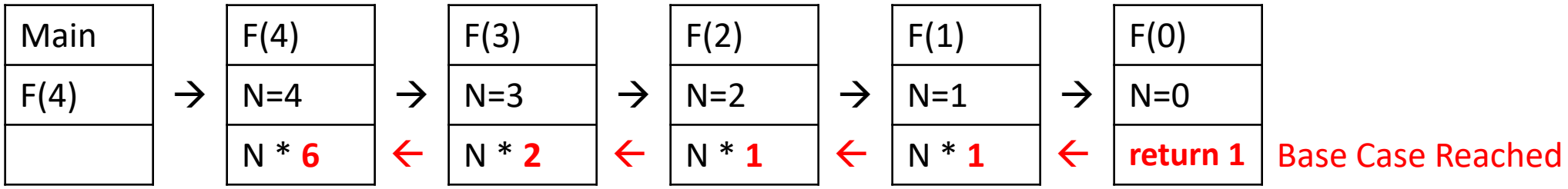


SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method

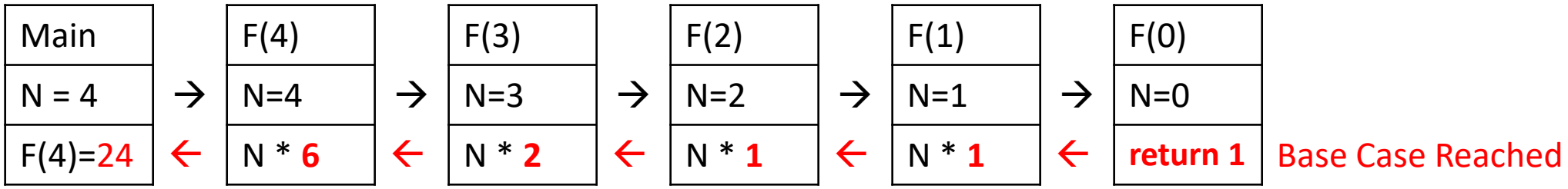

Main
F(4)
F(4)
N=4
N * f(3)
F(3)
N=3
N * 2
F(2)
N=2
N * 1
F(1)
N=1
N * 1
F(0)
N=0
return 1
Base Case Reached

Box Method



SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method



SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Factorial Iterative

```
unsigned int ifactorial( unsigned int n )
{
    int i;
    unsigned int ans = 1;
    for( i=1; i<=n; i++)
        ans *= i;
    return ans;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Summation

$$\sum_{i=0}^{n} i = 0 + 1 + \ldots + (n-2) + (n-1) + n$$

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Summation

- How to define in terms of the first n-1 terms

$$\sum_{i=0}^{n} i = n + \sum_{i=0}^{n-1} i$$

- What will serve as our base case?
  - N = 0, answer is 0

Summation Criteria

N = 0 answer is 0

N > 0 answer is $n + \sum_{i=0}^{n-1} i$

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Drawing it out

- Assume n = 6
  - Sum(6) = 6 + Sum(5)
  - Sum(5) = 5 + Sum(4)
  - Sum(4) = 4 + Sum(3)
  - Sum(3) = 3 + Sum(2)
  - Sum(2) = 2 + Sum(1)
  - Sum(1) = 1 + Sum(0)
  - Sum(0) = 0 ← Good, I know the answer

Substitute Going Back Up

- Assume n = 6
  - Sum(6) = 6 + Sum(5) = 6 + 15 = 21
  - Sum(5) = 5 + Sum(4) = 5 + 10 = 15
  - Sum(4) = 4 + Sum(3) = 4 + 6 = 10
  - Sum(3) = 3 + Sum(2) = 3 + 3 = 6
  - Sum(2) = 2 + Sum(1) = 2 + 1 = 3
  - Sum(1) = 1 + Sum(0) = 1 + 0 = 1
  - Sum(0) = 0 ← Good, I know the answer

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Writing the function

```
unsigned int summation( unsigned int n )
{
    // write the base case first
    if( n == 0 )
        return 0;



}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Writing the function

```
unsigned int summation( unsigned int n )
{   int ans;
    // write the base case first
    if( n == 0 )
        return 0;

    // compute the answer
    ans = n + summation( n - 1 );
    return ans;

}
```

Stepping Through the Function

```
int main()
{
    int result;
    int n = 4;
    result = summation ( n );
    cout << "sum of 0 to " << n << " = "
         << result << endl;
    return 0;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main
? = S(4)	result
4	n

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)
? = S(4)	Result	
4	n	
4		n
? = N + S(3)		ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)
? = S(4)	Result		
4	n		
4		n	
? = n + S(3)		ans	
3			n
? = n + S(2)			ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2)
? = S(4)	Result			
4	n			
4		n		
? = n + S(3)		ans		
3			n	
? = n + S(2)			ans	
2				n
? = n + S(1)				ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2)	S(1)
? = S(4)	Result				
4	n				
4		n			
? = n + S(3)		ans			
3			n		
? = n + S(2)			ans		
2				n	
? = n + S(1)				ans	
1					n
? = n + S(0)					ans

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2)	S(1)	S(0)
? = S(4)	Result					
4	n					
4		n				
? = n + S(3)		ans				
3			n			
? = n + S(2)			ans			
2				n		
? = n + S(1)				ans		
1					n	
? = n + S(0)					ans	
0						n
						ans

Base Case Reached

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2)	S(1)	S(0) = 0
? = S(4)	Result					
4	n					
4		n				
? = n + S(3)		ans				
3			n			
? = n + S(2)			ans			
2				n		
? = n + S(1)				ans		
1					n	
n + S(0) = 1					ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2)	S(1) = 1
? = S(4)	Result				
4	n				
4		n			
? = n + S(3)		ans			
3			n		
? = n + S(2)			ans		
2				n	
n + S(1) = 3				ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3)	S(2) = 3
? = S(4)	Result			
4	n			
4		n		
? = n + S(3)		ans		
3			n	
n + S(2) = 6			ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4)	S(3) = 6
? = S(4)	Result		
4	n		
4		n	
n + S(3) = 10		ans	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Memory (remember the scoping rules, s = summation)

Memory	main	S(4) = 10
= S(4) = 10	Result	
4	n	

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

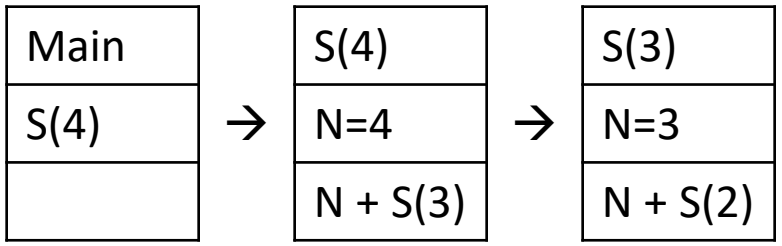
Box Method

Main
S(4)

→

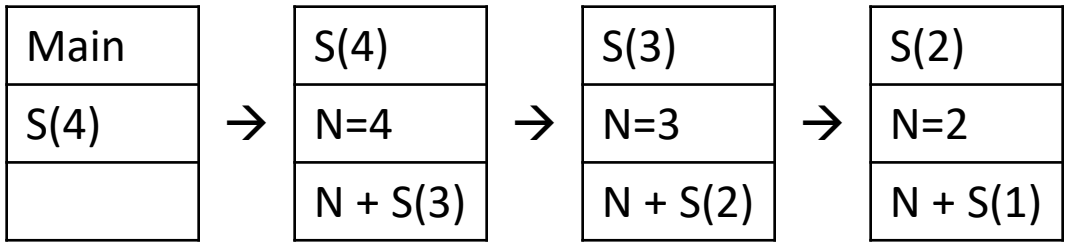
S(4)
N=4
N + S(3)

Box Method



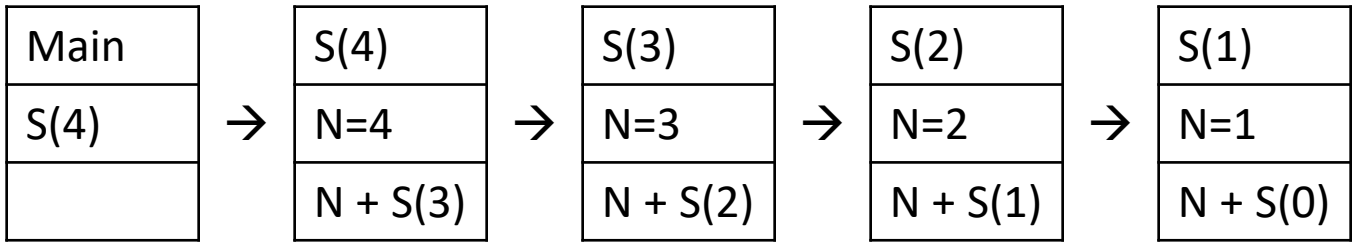
SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Box Method

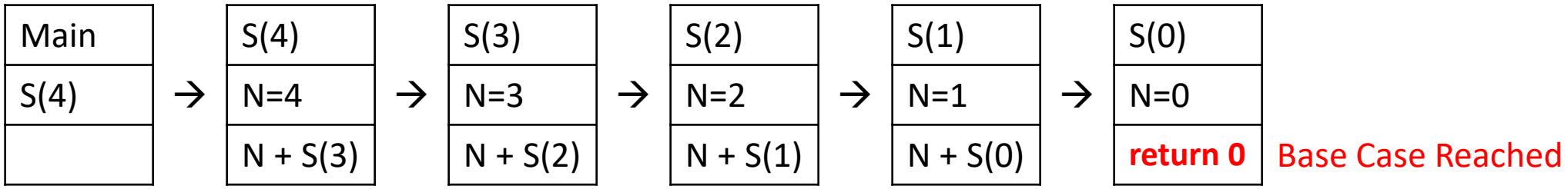


SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

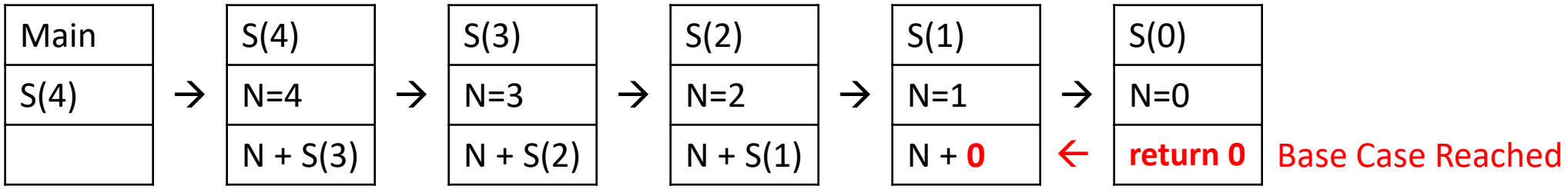
Box Method



Box Method

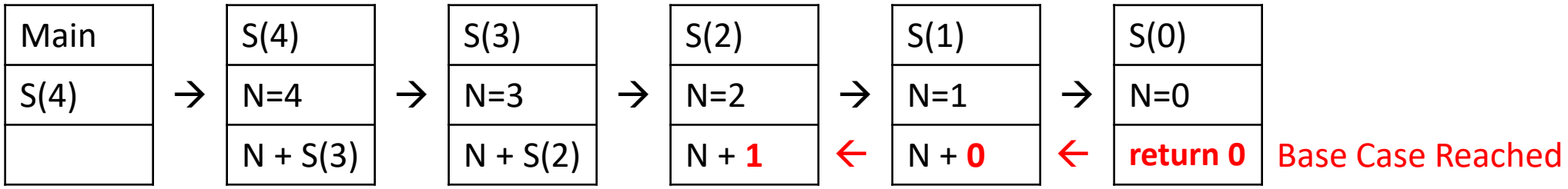

Main
S(4)
S(4)
N=4
N + S(3)
S(3)
N=3
N + S(2)
S(2)
N=2
N + S(1)
S(1)
N=1
N + S(0)
S(0)
N=0
return 0
Base Case Reached

Box Method



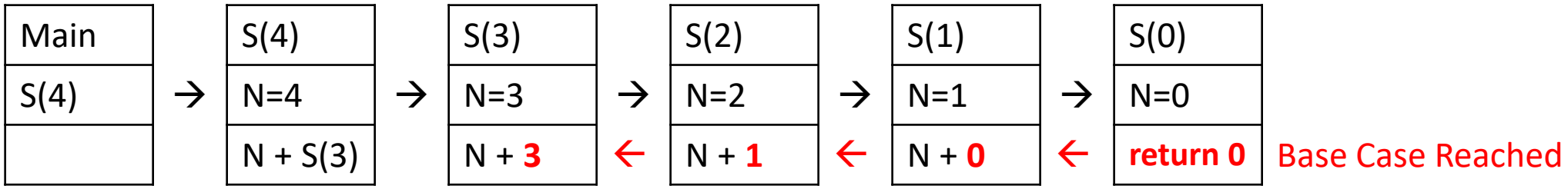
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method



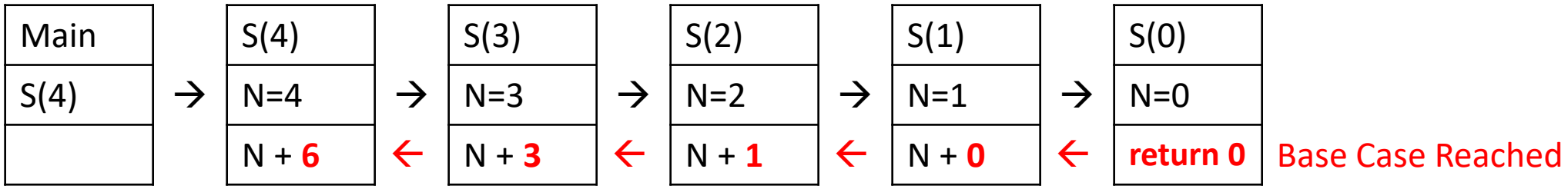
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method



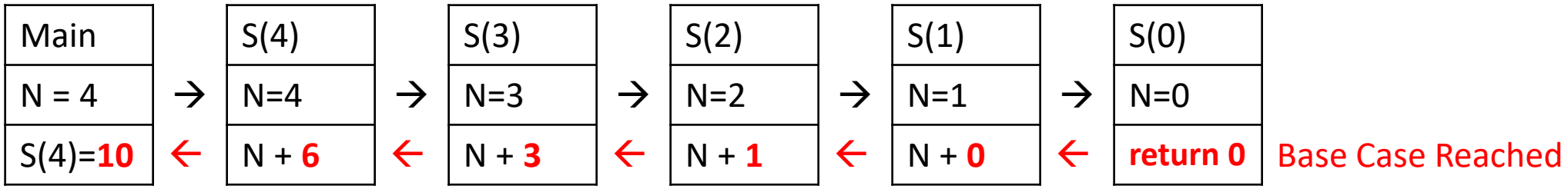
SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Box Method



SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Box Method



SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Summation Iterative

```
unsigned int isummation( unsigned int n )
{
    int i;
    unsigned int ans = 0;
    for( i=0; i<=n; i++)
        ans += i;
    return ans;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

You Try It at Home

- Given two integer number a and b where a is less than b and a is greater than or equal to 0. Sum the range from a to b.

Ie a = 3, b = 7

Sum(3,7) = 3 + 4 + 5 + 6 + 7 =25

Careful, Sum(3,3) = 3

To get started:

```
unsigned int sumRange(unsigned int a, unsigned int b );
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY