CSC215

Math and Computer Science

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A Legacy of Excellence

Recursion – Print C String out backwards

- Method 1
  - Move an index forward accessing each character in the array
  - If we reach the end of the array, quit.
  - As we back out we will print each character
  - For each function call
    - if the character at the i^th^ position is the '\0' we are done
    - Otherwise: Print the remaining characters and then print the i^th^ character

Method 1 Criteria

• cstr[i] == '\0'	done, nothing to print
• cstr[i] != '\0'	print remaining characters out backwards and then print cstr[i]

Writing the function

```
void sBackMethod1( char *str, int loc )
{
    // write the base case first
    if( str[loc] == '\0' )
        return;


}
```

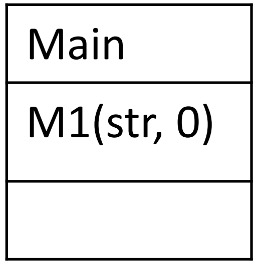
Writing the function

```
void sBackMethod1( char *str, int loc )
{
    // write the base case first
    if( str[loc] == '\0' )
        return;

    // print the remaining characters out then
    // print this character
    sBackMethod1( str, loc+1 );
    cout << str[loc];
}
```

Stepping through the function

```c
int main(int argc, char **argv)
{
    int i;
    char s[100] = "hello";

    sBackMethod1( s, 0 );
    return 0;
}
```

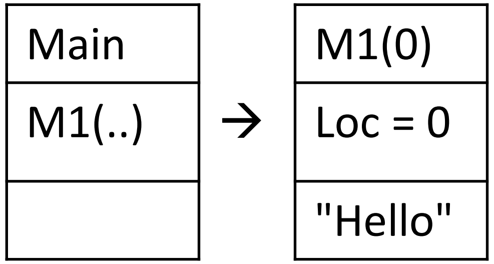
Box Method



Main

M1(str, 0)

Box Method



Main

M1(..)

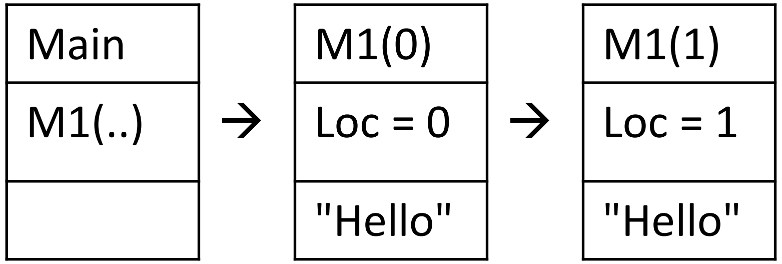
→

M1(0)

Loc = 0

"Hello"

Box Method



Main

M1(..)

→

M1(0)

Loc = 0

"Hello"

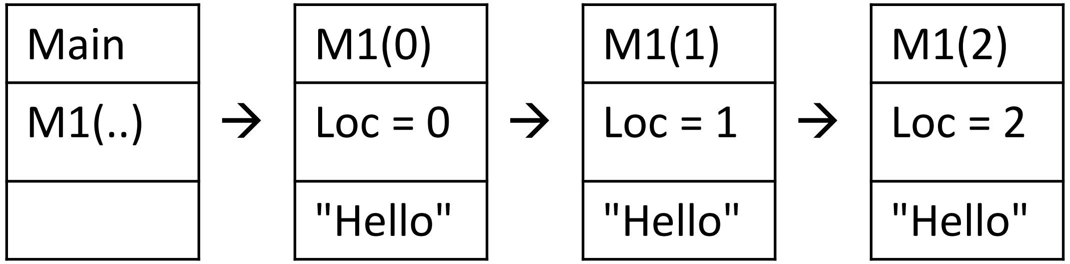
→

M1(1)

Loc = 1

"Hello"

Box Method



Main

M1(..)

→

M1(0)

Loc = 0

"Hello"

→

M1(1)

Loc = 1

"Hello"

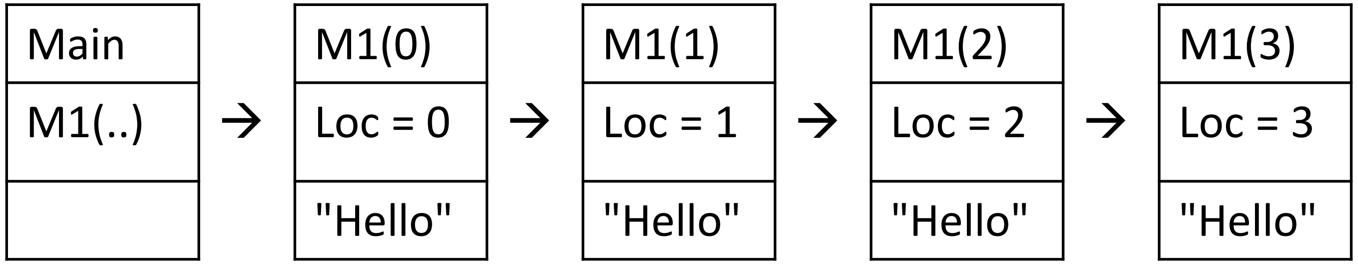
→

M1(2)

Loc = 2

"Hello"

Box Method



Main
M1(..) →

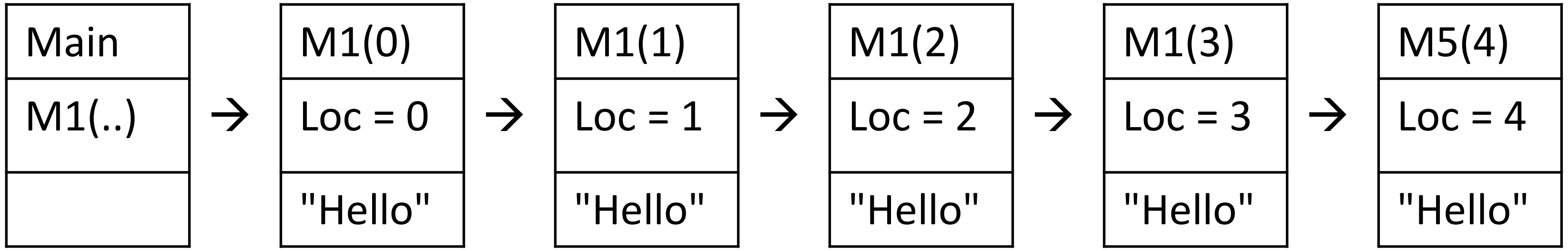
M1(0)
Loc = 0
"Hello" →

M1(1)
Loc = 1
"Hello" →

M1(2)
Loc = 2
"Hello" →

M1(3)
Loc = 3
"Hello"

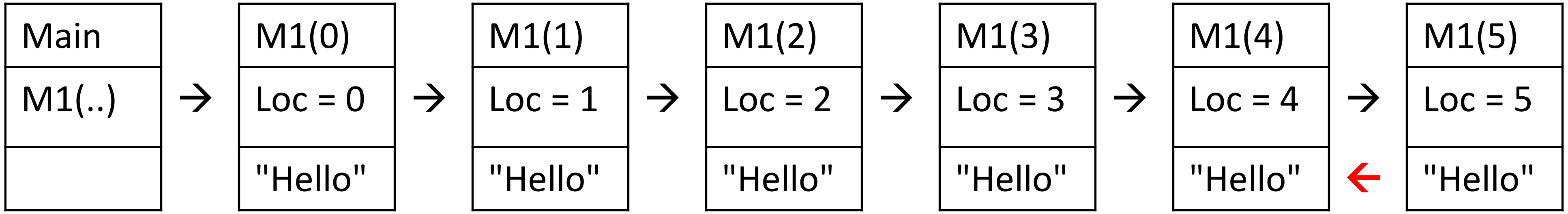
Box Method



Main → M1(0) → M1(1) → M1(2) → M1(3) → M5(4)

Main	M1(0)	M1(1)	M1(2)	M1(3)	M5(4)
M1(..)	Loc = 0	Loc = 1	Loc = 2	Loc = 3	Loc = 4
	"Hello"	"Hello"	"Hello"	"Hello"	"Hello"

Box Method



Main
M1(..)

M1(0)
Loc = 0
"Hello"

M1(1)
Loc = 1
"Hello"

M1(2)
Loc = 2
"Hello"

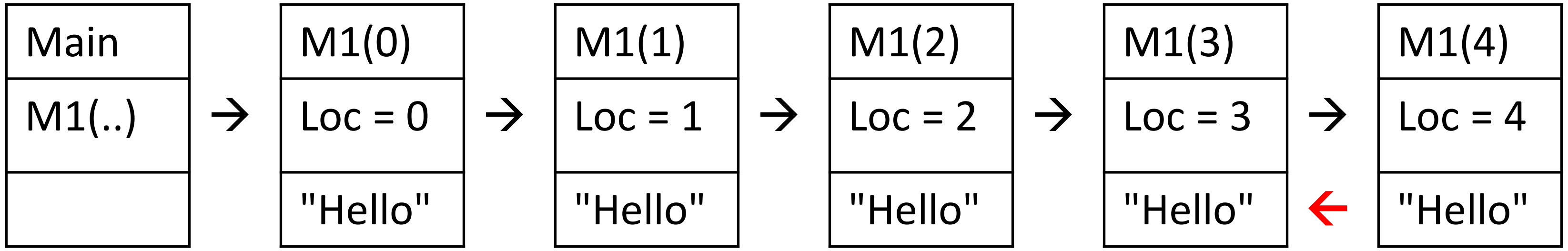
M1(3)
Loc = 3
"Hello"

M1(4)
Loc = 4
"Hello"

M1(5)
Loc = 5
"Hello"

Base case: print nothing and return

Box Method



Main → M1(0) → M1(1) → M1(2) → M1(3) → M1(4)

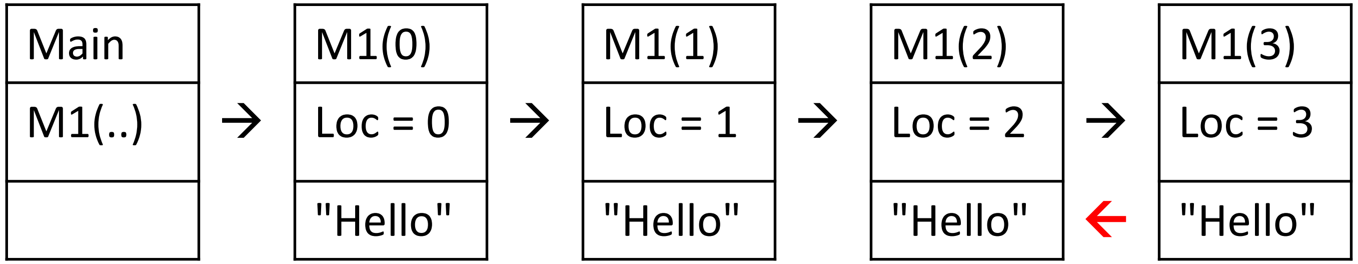
M1(..) → Loc = 0 → Loc = 1 → Loc = 2 → Loc = 3 → Loc = 4

"Hello" "Hello" "Hello" "Hello" ← "Hello"

Print the 4th index location and return

Console output:
o

Box Method



Main
M1(..)

→

M1(0)
Loc = 0
"Hello"

→

M1(1)
Loc = 1
"Hello"

→

M1(2)
Loc = 2
"Hello"

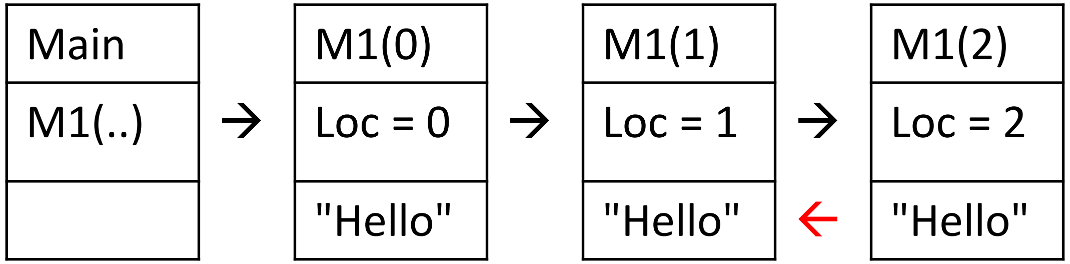
→ ←

M1(3)
Loc = 3
"Hello"

Print the 3th index location and return

Console output:
ol

Box Method



Main

M1(..)

→

M1(0)

Loc = 0

"Hello"

→

M1(1)

Loc = 1

"Hello"

→ ←

M1(2)

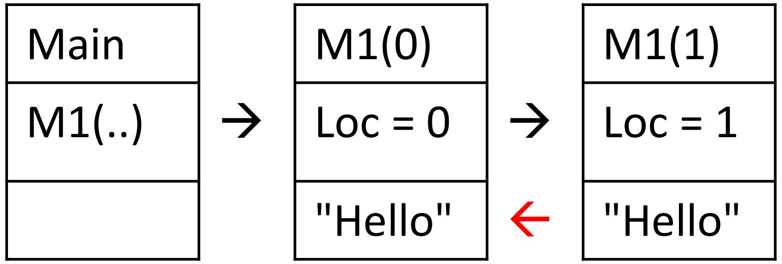
Loc = 2

"Hello"

Print the 2th index location and return

Console output:
oll

Box Method



Main

M1(..)

→

M1(0)

Loc = 0

"Hello"

→

←

M1(1)

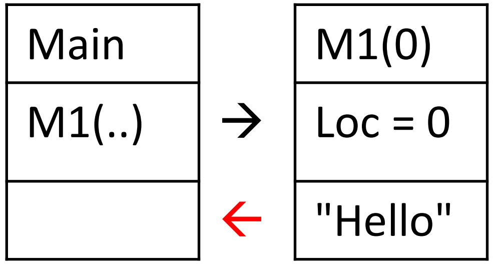
Loc = 1

"Hello"

Print the 1th index location and return

Console output:
olle

Box Method



Main

M1(..)

→

←

M1(0)

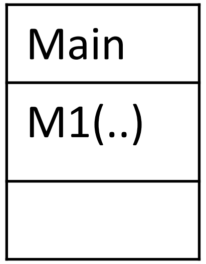
Loc = 0

"Hello"

Print the 0th index location and return

Console output:
olleH

Box Method



Main

M1(..)

Print the 0th index location and return

Console output:

```
olleH
```

Recursion – Print C String out backwards

- Method 2
  - Move an index backwards accessing each character in the array printing the character to the screen.
  - If the index becomes negative, we are done.
  - For each function call
    - if the i^{th} position is negative, return
    - Otherwise: print the i^{th} character then print the remaining characters.

Method 2 Criteria

- Ith position < 0
- Ith position >=0

done, nothing to print

print the character at the ith position and the print out the remaining (i-1) characters

Writing the function

```
void sBackMethod2( char *str, int loc )
{
    // write the base case first
    if( loc < 0 )
        return;

}
```

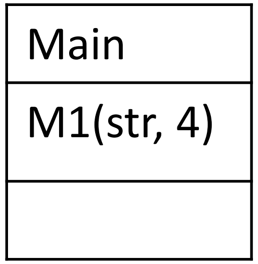
Writing the function

```
void sBackMethod2( char *str, int loc )
{
    // write the base case first
    if( str[loc] < 0 )
        return;

    // print this character
    // print the remaining characters out then
    cout << str[loc];
    sBackMethod2( str, loc-1 );
}
```

Stepping through the function

```
int main(int argc, char **argv)
{
    int i;
    char s[100] = "hello";

    sBackMethod2( s, 4 );
    return 0;
}
```

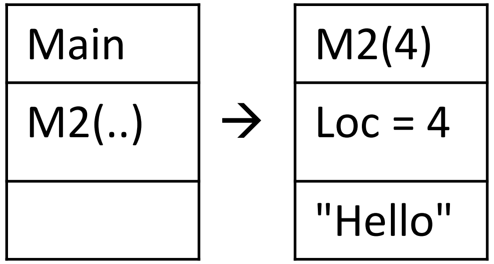
Box Method



Main

M1(str, 4)

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

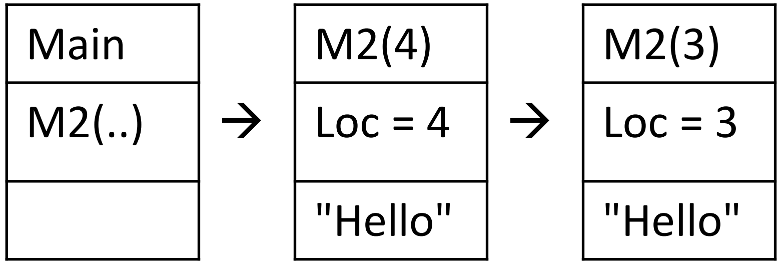
"Hello"

Print the 4th index location and then make a function call to print the others

Console output:

```
o
```

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→

M2(3)

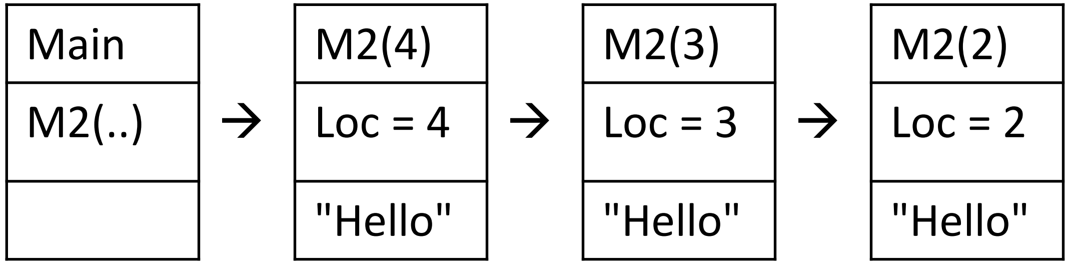
Loc = 3

"Hello"

Print the 3th index location and then make a function call to print the others

Console output:
ol

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→

M2(3)

Loc = 3

"Hello"

→

M2(2)

Loc = 2

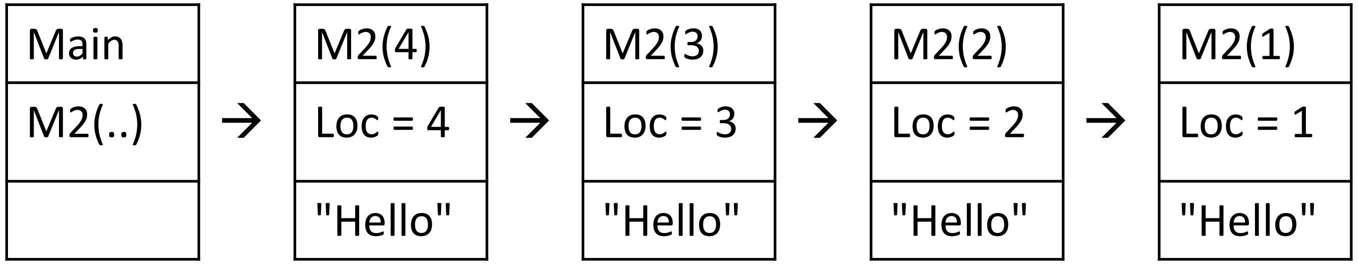
"Hello"

Print the 2th index location and then make a function call to print the others

Console output:

```
oll
```

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→

M2(3)

Loc = 3

"Hello"

→

M2(2)

Loc = 2

"Hello"

→

M2(1)

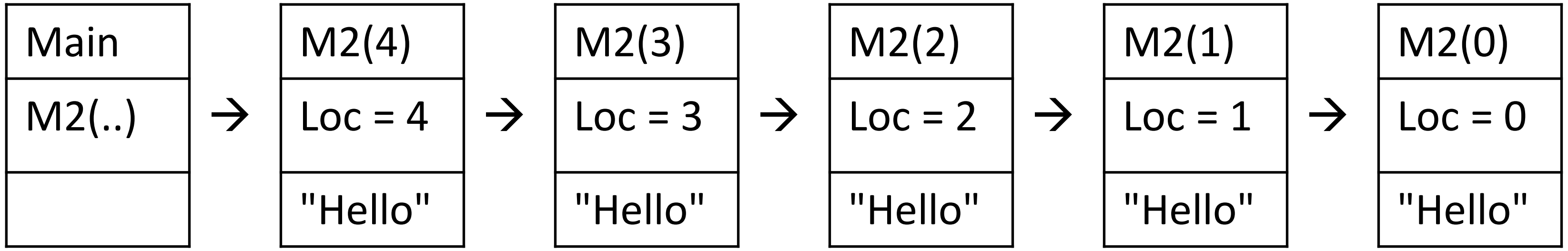
Loc = 1

"Hello"

Print the 1st index location and then make a function call to print the others

Console output:
olle

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→

M2(3)

Loc = 3

"Hello"

→

M2(2)

Loc = 2

"Hello"

→

M2(1)

Loc = 1

"Hello"

→

M2(0)

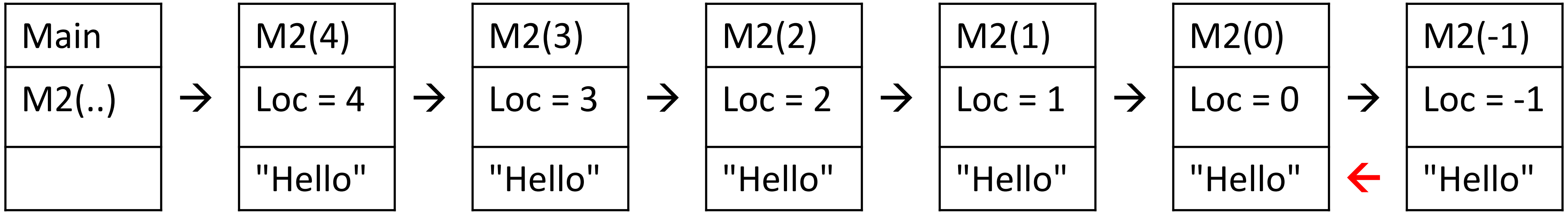
Loc = 0

"Hello"

Print the 0th index location and then make a function call to print the others

Console output:
olleH

Box Method



Main | M2(..)

→

M2(4) | Loc = 4 | "Hello"

→

M2(3) | Loc = 3 | "Hello"

→

M2(2) | Loc = 2 | "Hello"

→

M2(1) | Loc = 1 | "Hello"

→

M2(0) | Loc = 0 | "Hello"

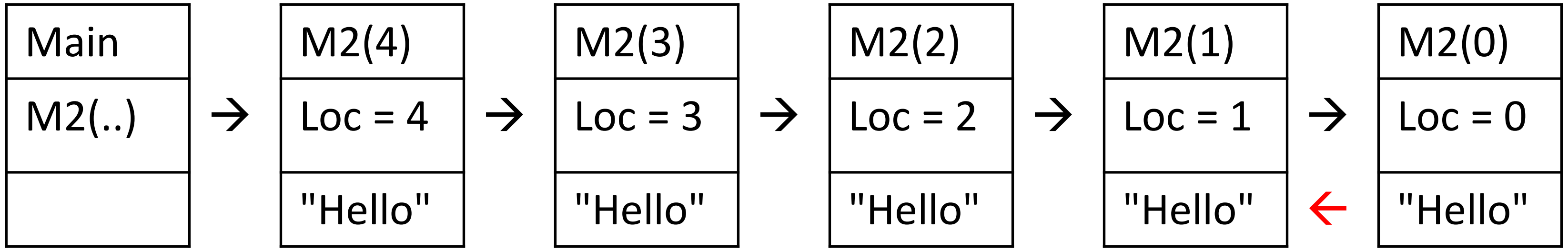
→ ←

M2(-1) | Loc = -1 | "Hello"

Base case: print nothing and return

Console output:
olleH

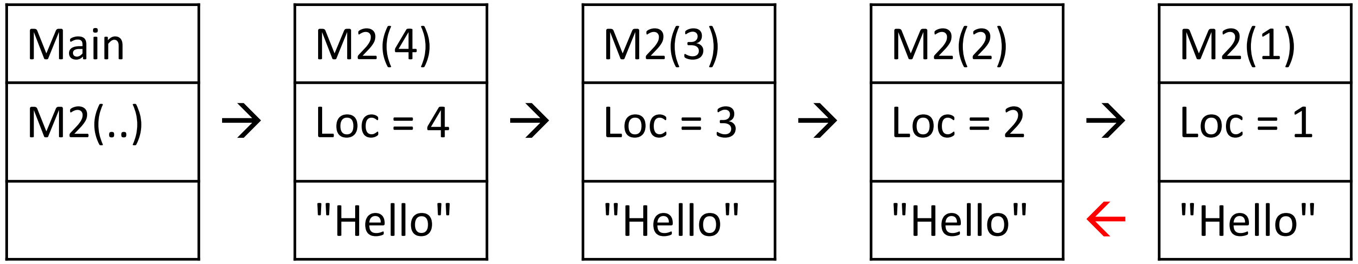
Box Method



Main		M2(4)		M2(3)		M2(2)		M2(1)		M2(0)
M2(..)	→	Loc = 4	→	Loc = 3	→	Loc = 2	→	Loc = 1	→	Loc = 0
		"Hello"		"Hello"		"Hello"		"Hello"	←	"Hello"

Console output:
olleH

Box Method



Main
M2(..)

→

M2(4)
Loc = 4
"Hello"

→

M2(3)
Loc = 3
"Hello"

→

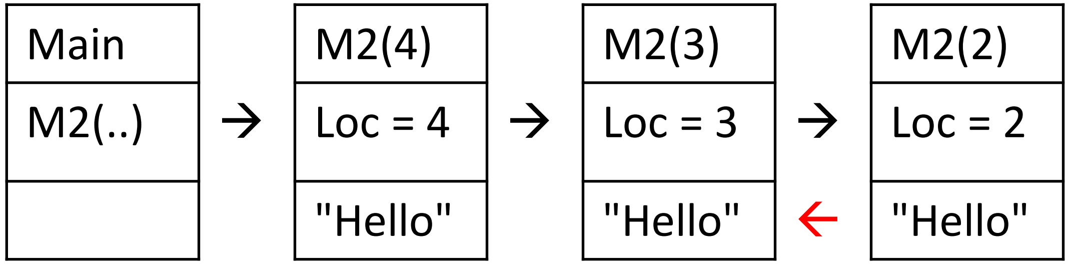
M2(2)
Loc = 2
"Hello"

→ ←

M2(1)
Loc = 1
"Hello"

Console output:
olleH

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→

M2(3)

Loc = 3

"Hello"

→

←

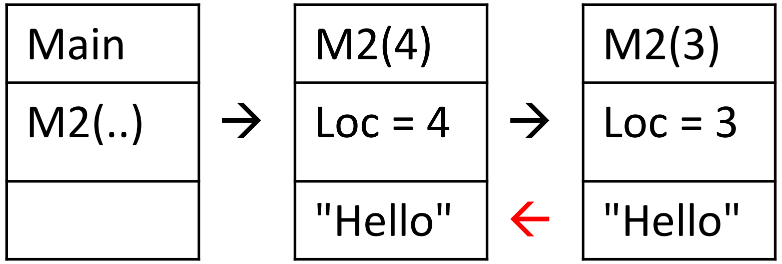
M2(2)

Loc = 2

"Hello"

Console output:
olleH

Box Method



Main

M2(..)

→

M2(4)

Loc = 4

"Hello"

→ ←

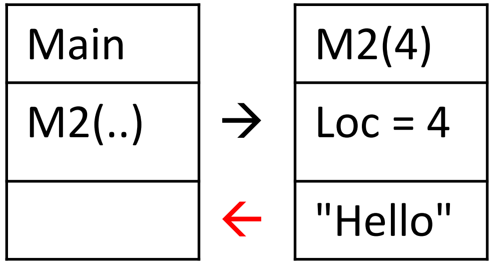
M2(3)

Loc = 3

"Hello"

Console output:
olleH

Box Method



Main

M2(..) →

←

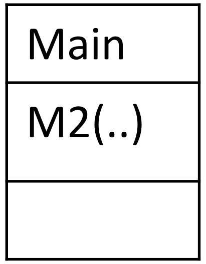
M2(4)

Loc = 4

"Hello"

Console output:
olleH

Box Method



Main

M2(..)

Back where the recursive function was first called

Console output:
olleH

Iterative Methods

```
int i;
char str[100] = "hello";
int len = strlen(str) - 1;

for( i=len; i>=0; i--)
    cout << str[i];
cout << endl;
```

Iterative Method – Bad but works

```
int i;
char str[100] = "hello";
int len = strlen(str);

reverse(str, str + len );
cout << str << endl;
reverse(str, str + len );
```

Iterative Method C++ String

```
int i;
string str = "Hello";
string::reverse_iterator it;

for( it= str.rbegin(); it != str.rend(); it++)
    cout << *it;
cout << endl;
```

Iterative Method C++ String - bad

```
int i;
string str = "Hello";

reverse( str.begin(), str.end());
cout << str << endl;
reverse( str.begin(), str.end());
```

Try These Recursively for Practice

- rstrcpy – recursively copy from one c string to another c string using indexes
- rstrlower – recursively convert a string from to all lower case
- rstrupper – recursively convert a string from to all Uppercase
- Rinvert – recursively convert characters that were uppercase to lower case and characters that were lower case to uppercase.

Try These Recursively for Practice

- rcharset – recursively set n number of characters to a character passed in. rcharset(str, ‘p’, 10) would copy 10 p’s to the str string.
- rstrcmp – recursively compare 2 c strings to each other using indexs to move from position to position
  - Return 0 if the 2 strings are equal
  - Return a number < 0 if str1 comes before str2
  - Return a number > 0 if str1 comes after str2

```
if( rstrcmp( str1, str2, 0) == 0 )
    cout << “The strings are equal” << endl;
```

SOUTH DAKOTA
M
SCHOOL OF MINES
& TECHNOLOGY