CS 240 LAB 4

Finding Factorial in Recursive Way

Recursive Definition of a Function

To define a function recursively, we should specify

- A rule for constructing new elements of function those can be specified with the function itself (recursive case)
- One or more initial elements of function (base case)
- Recursive calls must move on base condition

An Example -> Factorial:

- 0! = 1, 1! = 1 base case
- n! = n * (n-1)! for n > 1 recursive case

- Thus, 3 = 3 * 2!
- = 3 * 2 * 1!
- = 3 * 2 * 1 * 0!
- = 3 * 2 * 1 * 1 (= 6)

High Level C Code

```
int factorial(int n) {
        if (n == 0 || n == 1)  return 1;
        return n * factorial(n-1);
}
```

In another saying;

```
int factorial(int n) {
        if (n == 0 || n == 1)  return 1;
        int from_stack = factorial(n-1);
        return n * stack_return;
}
```

How does recursion work inside a computer?

factorial(3)

- int factorial(int n) {

-> if (n == 0 || n == 1) return 1;

    int from_stack = factorial(n-1);

    return n * from_stack ;

- }

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	

factorial(3)

- int factorial(int n) {
  if (n == 0 || n == 1) return 1;
->int from_stack = factorial(n-1);
  return n * from_stack ;
- }

Call factorial(2)

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	

factorial(2)

- int factorial(int n) {

-> if (n == 0 || n == 1) return 1;

    int from_stack = factorial(n-1);

    return n * from_stack ;

- }

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	
parameter n	303	2
result return address	304	302
return value of factorial(n-1)	305	

factorial(2)

```
• int factorial(int n) {
    if (n == 0 || n == 1)  return 1;
-> int from_stack = factorial(n-1);
    return n * from_stack ;
• }
```

Call factorial(1)

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	
parameter n	303	2
result return address	304	302
return value of factorial(n-1)	305	

factorial(1)

- int factorial(int n) {

-> if (n == 0 || n == 1) return 1;

    int from_stack = factorial(n-1);

    return n * from_stack ;

- }

STACK UNWIND TIME!

Return to factorial(2)

with 1

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	
parameter n	303	2
result return address	304	302
return value of factorial(n-1)	305	1
parameter n	306	1
result return address	307	305

factorial(2)

```
• int factorial(int n) {
    if (n == 0 || n == 1)  return 1;
    int from_stack = factorial(n-1);
 ->return n * from_stack ;
• }
```

Return to factorial(3)

with 2 * 1

	Address	Value
Final Result	20	
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	2
parameter n	303	2
result return address	304	302
return value of factorial(n-1)	305	1
parameter n	306	1
result return address	307	305

factorial(3)

```
• int factorial(int n) {
    if (n == 0 || n == 1)  return 1;
    int from_stack = factorial(n-1);
 ->return n * from_stack ;
• }
```

Return to main

with 3 * 2

	Address	Value
Final Result	20	6
parameter n	300	3
result return address	301	20
return value of factorial(n-1)	302	2
parameter n	303	2
result return address	304	302
return value of factorial(n-1)	305	1
parameter n	306	1
result return address	307	305

XYZ:

if(.....) goto ABC

goto XYZ

ABC:

if(.....) goto END

goto ABC

END:

goto END

Good Luck ☺