Recursion in C

Consider a function

```
#include <stdio.h>
void callMe(){

        printf("Hello World\n");
}
void main(){
        callMe();
}
```

Output

Hello World

Consider a function

```
#include <stdio.h>
void callMe(){

        printf("Hello World\n");
        callMe();
}
void main(){
        callMe();
}
```

Output

```
Hello World
Hello World
Hello World
….
…
```

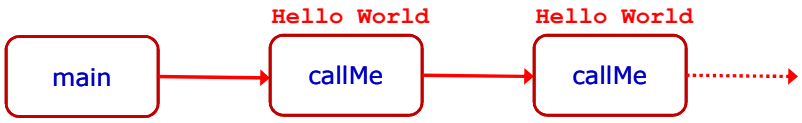

Consider a function

```
#include <stdio.h>
void callMe(){

        printf("Hello World\n");
        callMe();
}
void main(){
        callMe();
}
```

Output

```
Hello World
Hello World
Hello World
….
…
```



What is Recursion?

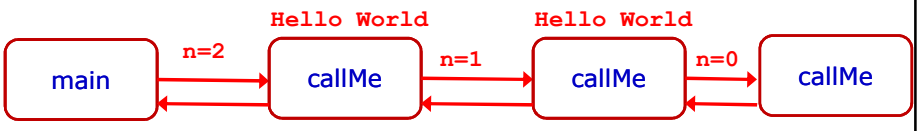
- When a function calls itself, the phenomenon is called RECURSION.
- Self-calling functions are often called recursive function.
- For successful recursion the self calling must be stopped.

How to stop?

```
#include <stdio.h>
void callMe(int n){
        if(n == 0) return;
        printf("Hello World\n");
        callMe(n-1);
}
void main(){
        callMe(2);
}
```

Output

```
Hello World
Hello World
```

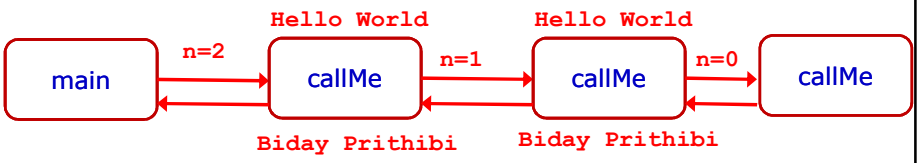


Another example

```
#include <stdio.h>
void callMe(int n){
        if(n == 0) return;
        printf("Hello World\n");
        callMe(n-1);
        printf("Biday Prithibi\n");
}
void main(){
        callMe(2);
}
```

```
Output

        Hello World
        Hello World
        Biday Prithibi
        Biday Prithibi
```



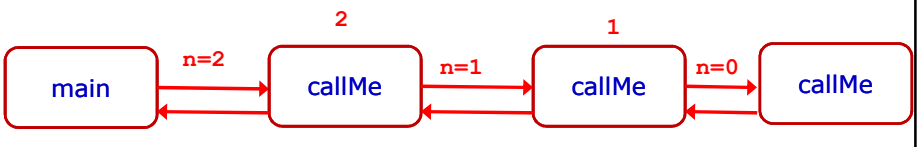
A Rahman Recursion 7

Printing numbers (downward)

```
#include <stdio.h>
void callMe(int n){
        if(n == 0) return;
        printf("%d\n",n);
        callMe(n-1);
}
void main(){
        callMe(2);
}
```

```
Output

        2
        1
```



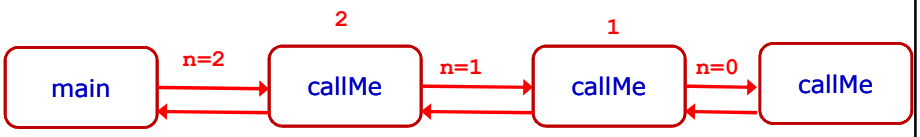
A Rahman Recursion 8

Printing numbers (upward)

```
#include <stdio.h>
void callMe(int n){
        if(n == 0) return;
        callMe(n-1);
        printf("%d\n",n);
}
void main(){
        callMe(2);
}
```

Output

```
1
2
```



We did some analysis of recursion. Now we will see some problem solving using recursion.

Recursion main logic

a) Find Recursive Definition (RD) of the problem
b) Find Base case where we do not need any recursion
c) Put Base case inside if block and recursive part under else block.

Problem 1

Write down a recursive function that will recursively compute summation of all natural numbers up to N. N will be a parameter to your function.

Solution:

sum(N-1)

sum(N) = 1 + 2 + 3 +…….(N-1) + N

Recursive Definition: sum(N) = N + sum(N-1)

Base Case: sum(1) = 1

Solution

a) Find Recursive Definition (RD) of the problem
b) Find Base case where we do not need any recursion
c) Put Base case inside if block and recursive part under else block.

Solution:

sum(N) = 1 + 2 + 3 +…….(N-1) + N

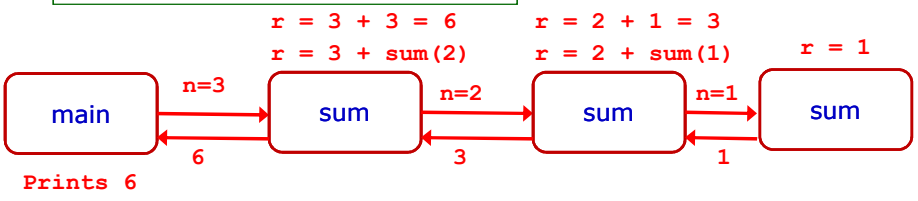
Recursive Definition: sum(N) = N + sum(N-1)
Base Case: sum(1) = 1

```
int sum(int n){
        int r;
        if(n == 1) r = 1;
        else r = n + sum(n-1);
        return r;
}
void main(){
        int x = sum(3);
        printf("%d",x);
}
```


Analysis

```
int sum(int n){
        int r;
        if(n == 1) r = 1;
        else r = n + sum(n-1);
        return r;
}
void main(){
        int x = sum(3);
        printf("%d",x);
}
```

Solution: N = 3
sum(3) = 1 + 2 + 3 = 6



Problem 2

Write down a recursive function that will compute factorial of N recursively. N will be a parameter to your function.

Solution:

fact(N-1)

fact(N) = 1 × 2 × 3 × …….(N-1) × N

Recursive Definition: fact(N) = N × fact(N-1)

Base Case: fact(0) = 1

Solution

a) Find Recursive Definition (RD) of the problem
b) Find Base case where we do not need any recursion
c) Put Base case inside if block and recursive part under else block.

Solution:

fact(N) = 1 × 2 × 3 × …….(N-1) × N

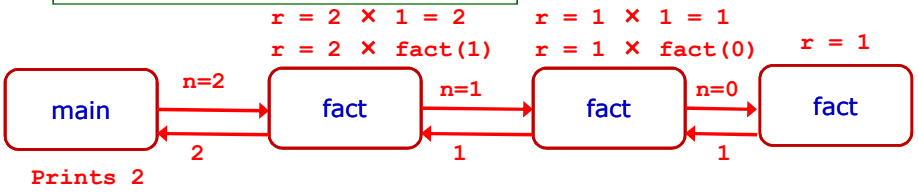
Recursive Definition: fact(N) = N × fact(N-1)
Base Case: fact(0) = 1

```
int fact(int n){
        int r;
        if(n == 0) r = 1;
        else r = n × fact(n-1);
        return r;
}
void main(){
        int x = fact(2);
        printf("%d",x);
}
```

Analysis

```
int fact(int n){
        int r;
        if(n == 0) r = 1;
        else r = n × fact(n-1);
        return r;
}
void main(){
        int x = fact(2);
        printf("%d",x);
}
```

Solution: N = 2
fact(2) = 1 × 2 = 2



Problem 3

Write down a recursive function that will compute x^N recursively. Both x, N will be a parameter to your function.

Solution: x^{N-1}

power(x, N) = x × x × x × x × …….x × x

Recursive Definition: power(x, N) = x × power(x, N-1)

Base Case: power(x, 0) = 1

Solution

a) Find Recursive Definition (RD) of the problem
b) Find Base case where we do not need any recursion
c) Put Base case inside if block and recursive part under else block.

Solution:

power(x, N) = x × x × x × …….x × x

Recursive Definition: power(x, N) = x × power(x, N-1)
Base Case: power(x,0) = 1

```
int power(int x, int n){
        int r;
        if(n == 0) r = 1;
        else r = x × power(x,n-1);
        return r;
}
void main(){
        int y = power(3,2);
        printf("%d",y);
}
```


Analysis

```
int power(int x, int n){
        int r;
        if(n == 0) r = 1;
        else r = x × power(x,n-1);
        return r;
}
void main(){
        int y = power(3,2);
        printf("%d",y);
}
```

Solution: x = 3, N = 2
power(3, 2) = 3 × 3 = 9