Recursive Functions

Dr. H. İrem Türkmen

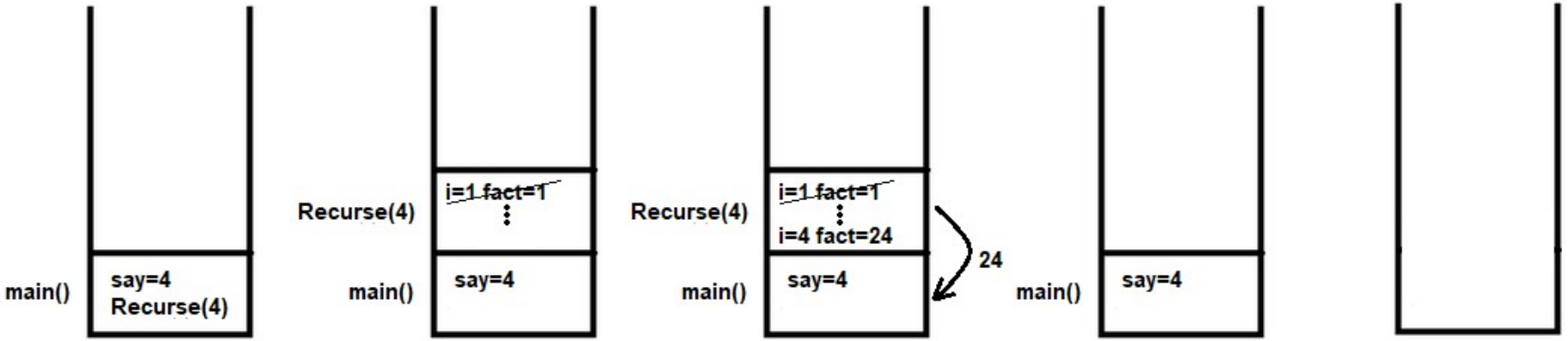
Recursive Functions

- A **recursive function** can be defined as a routine that calls itself directly
- Recursion can reduce time complexity.
- Recursion adds clarity and reduces the time needed to write and debug code.
- Requires more memory due to function call overhead.

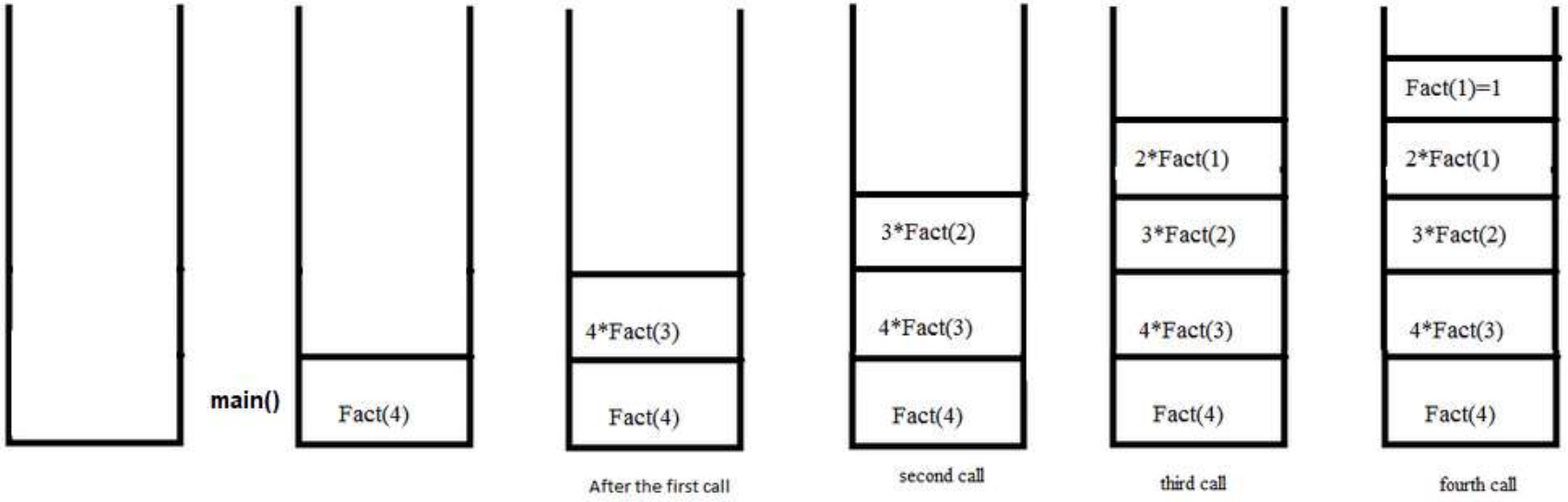
Factorial: n!=n*(n-1)! F(n)=n*F(n-1)

```
#include <stdio.h>
int factRecurseInf(int n)
{
        printf("processing..");
        return n*factRecurseInf(n-1);
}

int factRecurse(int n)
{
        if (n==1) return 1;
        else
                return n*factRecurse(n-1);
}

int main( )
{
        printf ( "%d\n", factRecurse(4) );
//      printf ( "%d\n", factRecurseInf(4) );
        return 0;
}
```

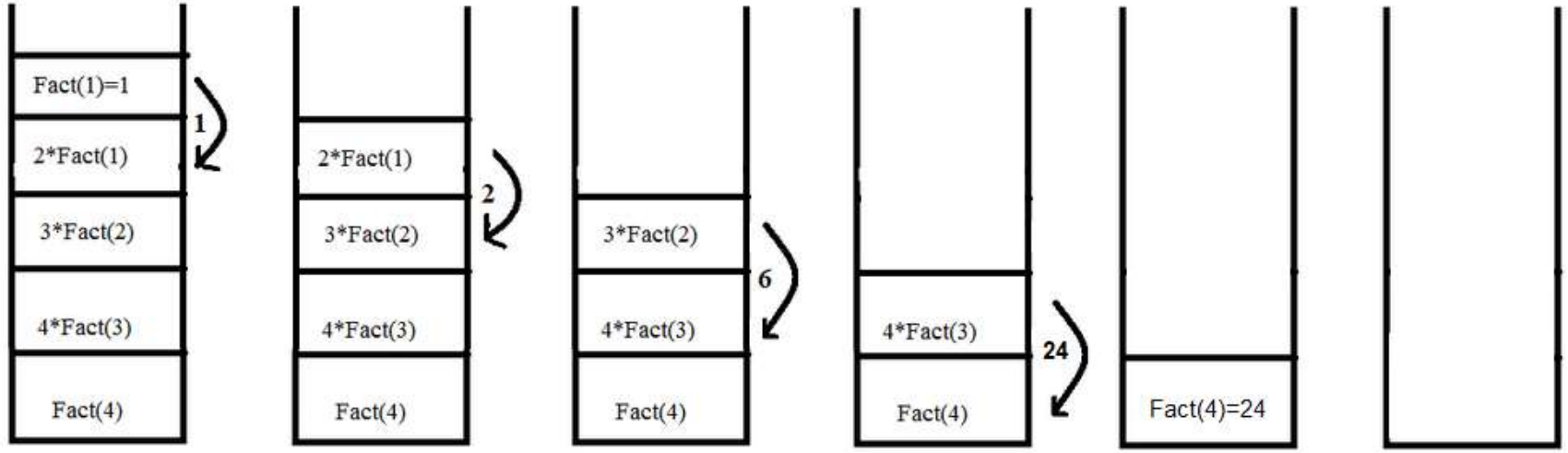
```
#include <stdio.h>
int Recurse(int n)
{
        int i,fact=1;
        for (i=1;i<=n;i++)
                fact=fact*i;
        return fact;
}
int main( )
{
        int say=4;
        printf("%d",Recurse(say));
        return 0;
}
```



When function call happens previous variables gets stored in stack



Returning values from base case to caller function




```
int factRecurse(int n)
{
  if (n==1)
    return 1;
  else
    return n*factRecurse(n-1);
}
int main( )
{
  printf ( "%d\n", factRecurse(4));
  return 0;
}
```

Exponentiation

- Avoid stack overflow !!
- What if factRecurse(-3) ??
- Exponentiation: $x^n = x * x^{n-1}$

```
int UsRecurse(int x, int n)
{
        if (n==1) return x;
        else
                return x*UsRecurse(x,n-1);
}
```

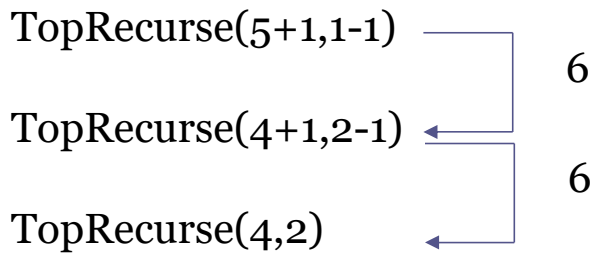
- M=UsRecurseV3(-4,2) ?
- M=UsRecurseV3(4,-2) ?

```
#include <stdio.h>
float UsRecurse(int x, int n)
{
	if (n==1) return x;
	else
		return x*UsRecurse(x,n-1);
}
float UsRecurseV2(int x, int n)
{
	if (n==0) return 1;
	else
		return 1.0/x*UsRecurseV2(x,n+1);
}
float UsRecurseV3(int x, int n)
{
	if (n==0) return 1;
	else
		if (n<0)	return 1.0/x*UsRecurseV3(x,n+1);
	return x*UsRecurse(x,n-1);
}
int main( )
{
	printf ( "%f\n", UsRecurseV3(4,-2) );
	return 0;
}
```

Calculate the sum of two numbers

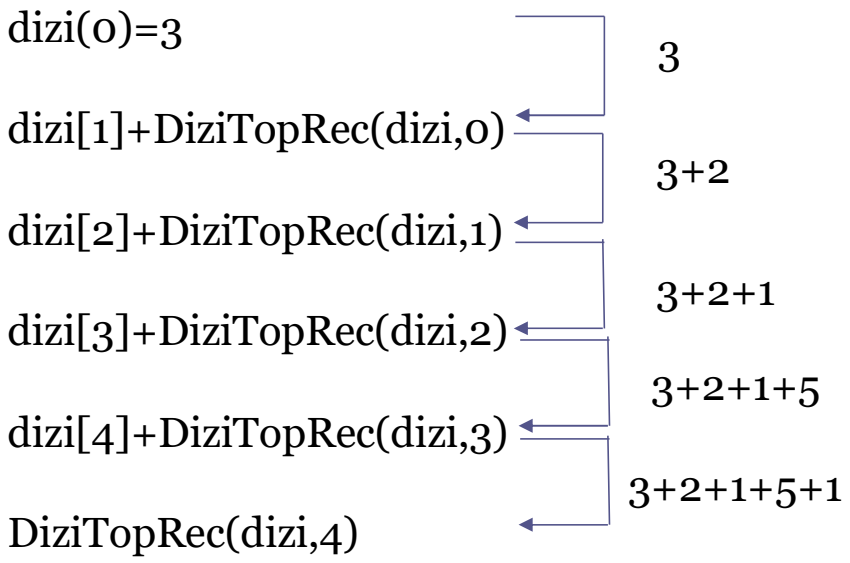
- What is the purpose? What if a and b are negative?

```
int TopRecurse(int a, int b)
{
        if (b==0)
                return a;
        else
                TopRecurse(a+1,b-1);
}
int main( )
{
        printf ( "%d\n", TopRecurse(4,2) );
        return 0;
}
```



Calculate the sum of array elements

```
int DiziTopRecurse(int* dizi, int b)
 {
        if (b==0) return dizi[b];
        else
          return dizi[b]+DiziTopRecurse(dizi,b-1);
}
int main( )
{
        int dizi[5]={3,2,1,5,1};
        printf ( "%d\n", DiziTopRecurse(dizi,4) );
        return 0;
}
```



Search an array element

```
int ElemanAraRecurse(int* dizi, int b, int x)
{
	if (b<0) return -1;
	else
		if (dizi[b]==x)
			return b;
		else
			return ElemanAraRecurse(dizi,b-1,x);
}
int main( )
{
	int dizi[5]={3,2,1,5,1};
	printf ( "%d\n", ElemanAraRecurse(dizi,4,7) );
	return    0;
}
```