Lesson 4: Recursion

Jiun-Long Huang
Department of Computer Science
National Chiao Tung University

Local Variables

- Local variables
  - Automatic storage duration
  - Block scope
- Global variables
  - Static storage duration
  - File scope
- C Tutor
  - http://www.pythontutor.com/c.html#mode=edit

Call By Value

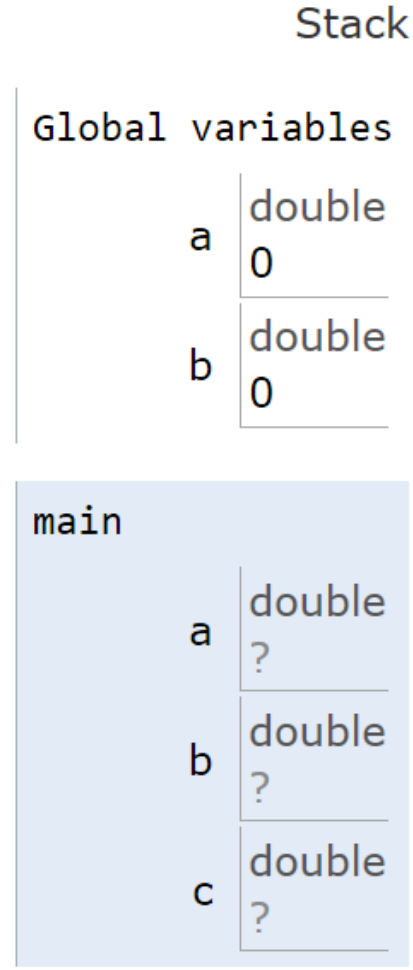
```
double a, b;
double average(double a, double b)
{
  return (a+b)/2;
}
int main(void)
{
  double a, b, c;
  a=1.0;
  b=2.0;
  c=average(a,b);
  return 0;
}
```

C (gcc 4.8, C11)

EXPERIMENTAL! known bugs/limitations

```
1  double a, b;
2  double average(double a, double b)
3  {
4    return (a+b)/2;
5  }
6  int main(void)
7  {
8    double a, b, c;
9    a=1.0;
10   b=2.0;
11   c=average(a,b);
12   return 0;
13 }
```

Edit this code

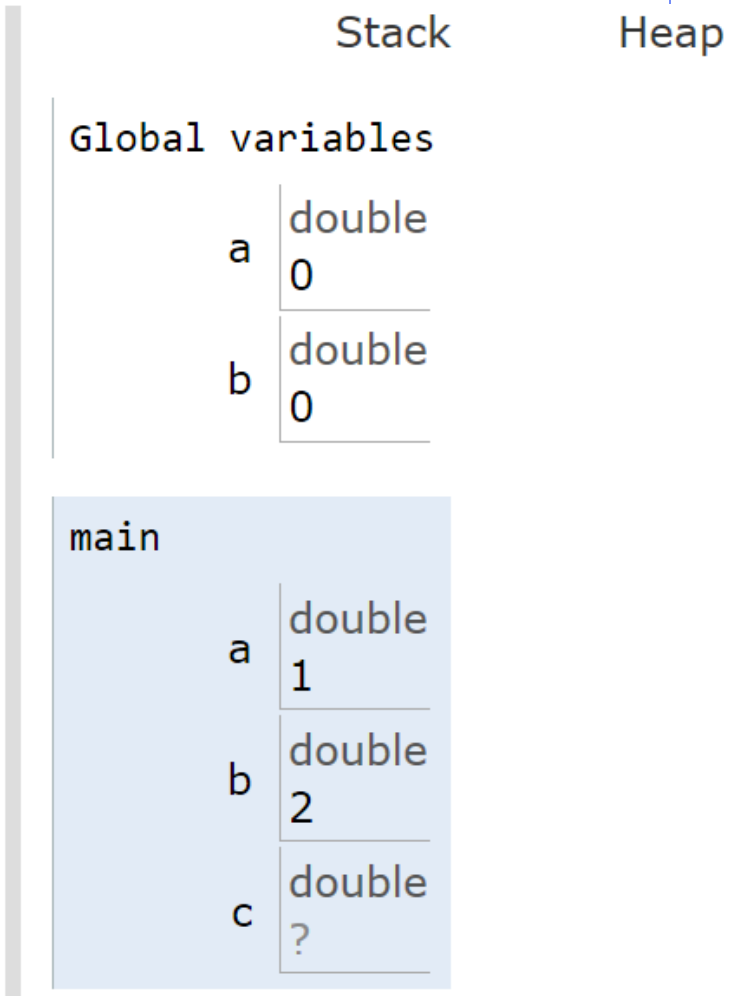


C (gcc 4.8, C11)

EXPERIMENTAL! known bugs/limitations

```
1  double a, b;
2  double average(double a, double b)
3  {
4    return (a+b)/2;
5  }
6  int main(void)
7  {
8    double a, b, c;
9    a=1.0;
10   b=2.0;
11   c=average(a,b);
12   return 0;
13 }
```

Edit this code



C (gcc 4.8, C11)
EXPERIMENTAL! known bugs/limitations

```
1  double a, b;
2  double average(double a, double b)
3  {
4    return (a+b)/2;
5  }
6  int main(void)
7  {
8    double a, b, c;
9    a=1.0;
10   b=2.0;
11   c=average(a,b);
12   return 0;
13 }
```

Edit this code

just executed
to execute

<< First | < Prev | Next > | Last >>

Step 5 of 9

Stack Heap

Global variables

a	double 0
b	double 0

main

a	double 1
b	double 2
c	double ?

average

a	double ?
b	double ?

C (gcc 4.8, C11)
EXPERIMENTAL! known bugs/limitations

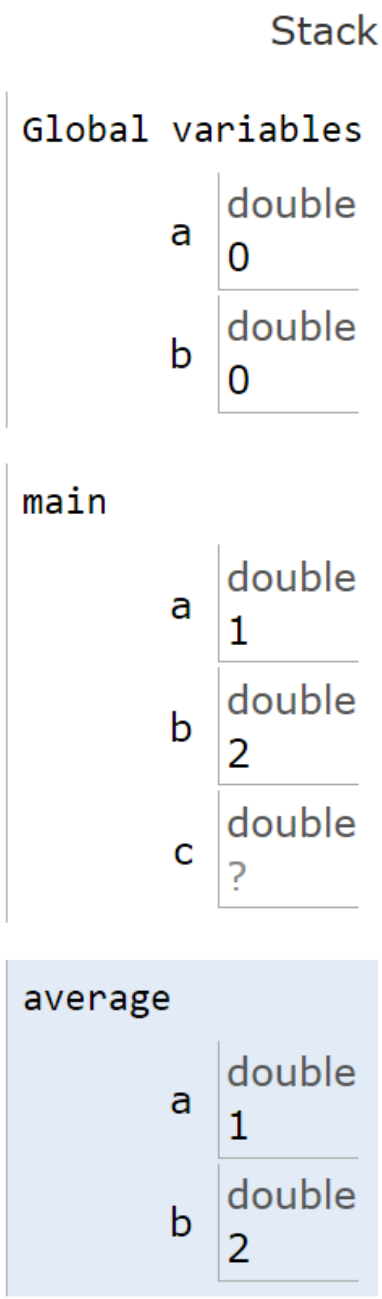
```
1  double a, b;
2  double average(double a, double b)
3  {
4    return (a+b)/2;
5  }
6  int main(void)
7  {
8    double a, b, c;
9    a=1.0;
10   b=2.0;
11   c=average(a,b);
12   return 0;
13 }
```

Edit this code

just executed
to execute

<< First | < Prev | Next > | Last >>

Step 6 of 9

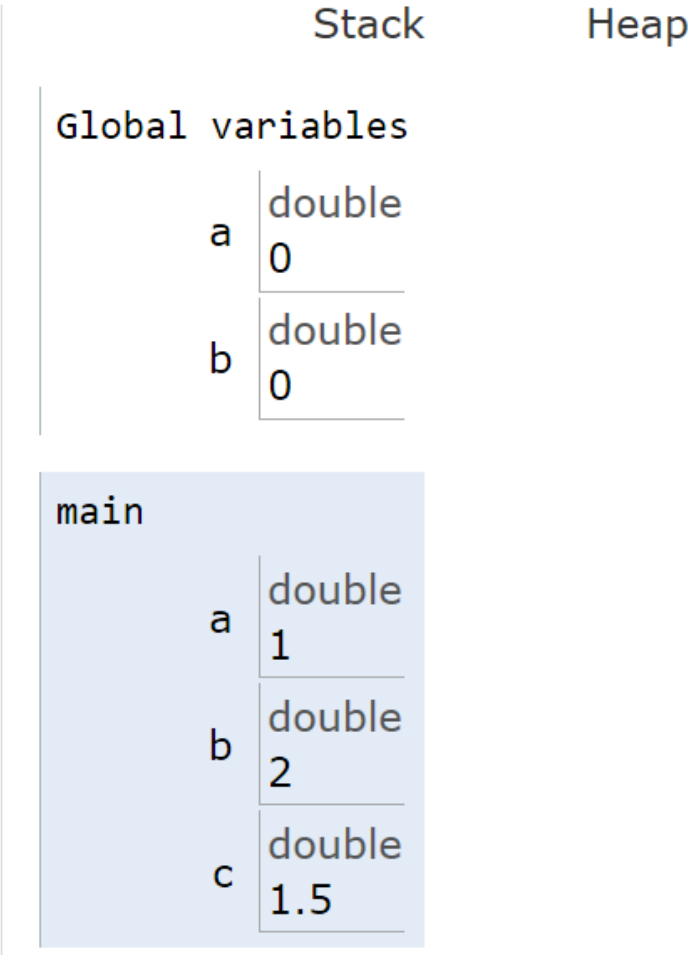


C (gcc 4.8, C11)

EXPERIMENTAL! known bugs/limitations

```
1  double a, b;
2  double average(double a, double b)
3  {
4    return (a+b)/2;
5  }
6  int main(void)
7  {
8    double a, b, c;
9    a=1.0;
10   b=2.0;
11   c=average(a,b);
12   return 0;
13 }
```

Edit this code



```c
#include <stdio.h>
int x=1, y=2; // global variables
int sum(int x, int y) // parameters
{
  int sum=0; // local variables
  sum=sum+x+y;
  x++;
  return sum;
}
int main(void)
{
  // call by value
  printf("%d\n",sum(x,y));
  int x=10, y=20; // local variables
  printf("%d\n",sum(x,y));
  return 0;
}
```

Initialize multiple times.

Result:
3
30

C (gcc 4.8, C11)
(known limitations)

```
 1  #include <stdio.h>
 2  int x=1, y=2; // global variables
 3  int sum(int x, int y) // parameters
 4  {
 5    int sum=0; // local variables
 6    sum=sum+x+y;
 7    x++;
 8    return sum;
 9  }
10  int main()
11  {
12    // call by value
13    printf("%d\n",sum(x,y));
14    int x=10, y=20; // local variables
15    printf("%d\n",sum(x,y));
16    return 0;
17  }
```

Print output (drag lo

Stack

Global variables

x	int 1
y	int 2

main

x	int ?
y	int ?

C (gcc 4.8, C11)
(known limitations)

```
1  #include <stdio.h>
2  int x=1, y=2; // global variables
3  int sum(int x, int y) // parameters
4  {
5    int sum=0; // local variables
6    sum=sum+x+y;
7    x++;
8    return sum;
9  }
10 int main()
11 {
12   // call by value
13   printf("%d\n",sum(x,y));
14   int x=10, y=20; // local variables
15   printf("%d\n",sum(x,y));
16   return 0;
17 }
```

Edit this code

at just executed
ne to execute

<< First | < Prev | Next > | Last >>

Step 3 of 17

e visualization (NEW!)

Print output (drag lo

Stack

Global variables

x	int 1
y	int 2

main

x	int ?
y	int ?

sum

x	int ?
y	int ?
sum	int ?

C (gcc 4.8, C11)
(known limitations)

```
1  #include <stdio.h>
2  int x=1, y=2; // global variables
3  int sum(int x, int y) // parameters
4  {
5    int sum=0; // local variables
6    sum=sum+x+y;
7    x++;
8    return sum;
9  }
10 int main()
11 {
12   // call by value
13   printf("%d\n",sum(x,y));
14   int x=10, y=20; // local variables
15   printf("%d\n",sum(x,y));
16   return 0;
17 }
```

Edit this code

t just executed
e to execute

<< First | < Prev | Next > | Last >>

Step 6 of 17

visualization (NEW!)

Print output (drag lov

Stack

Global variables

x	int 1
y	int 2

main

x	int ?
y	int ?

sum

x	int 1
y	int 2
sum	int 3

C (gcc 4.8, C11)
(known limitations)

```
1  #include <stdio.h>
2  int x=1, y=2; // global variables
3  int sum(int x, int y) // parameters
4  {
5    int sum=0; // local variables
6    sum=sum+x+y;
7    x++;
8    return sum;
9  }
10 int main()
11 {
12   // call by value
13   printf("%d\n",sum(x,y));
14   int x=10, y=20; // local variables
15   printf("%d\n",sum(x,y));
16   return 0;
17 }
```

Edit this code

ıt just executed
ıe to execute

<< First | < Prev | Next > | Last >>

Step 13 of 17

e visualization (NEW!)

Print output (drag lo

```
3
```

Stack

Global variables

x	int 1
y	int 2

main

x	int 10
y	int 20

sum

x	int 10
y	int 20
sum	int 0

Static Variables

- Static (local) variables
  - Static storage duration
  - Block scope

```c
#include <stdio.h>
int x=1,y=2; // global variables
int sum(int x, int y) // parameters
{
  static int sum=0; // static variables
  sum=sum+x+y;
  x++;
  return sum;
}
int main(void)
{
  // call by value
  printf("%d\n",sum(x,y));
  int x=10, y=20; // local variables
  printf("%d\n",sum(x,y));
  return 0;
}
```

Initialize once.

Result:
3
33

C (gcc 4.8, C11)
(known limitations)

```
1  #include <stdio.h>
2  int x=1,y=2; // global variables
3  int sum(int x, int y) // parameters
4  {
5    static int sum=0; // static variables
6    sum=sum+x+y;
7    x++;
8    return sum;
9  }
10 int main()
11 {
12   // call by value
13   printf("%d\n",sum(x,y));
14   int x=10, y=20; // local variables
15   printf("%d\n",sum(x,y));
16   return 0;
17 }
```

Edit this code

:hat just executed
line to execute

<< First | < Prev | Next > | Last >>

Step 3 of 15

ize visualization (NEW!)

Print output (drag lower right

Stack

Global variables

x	int	1
y	int	2

main

x	int	?
y	int	?

sum

x	int	?
y	int	?
sum (static 0x60104C)	int	0

C (gcc 4.8, C11)
(known limitations)

```
1  #include <stdio.h>
2  int x=1,y=2; // global variables
3  int sum(int x, int y) // parameters
4  {
5    static int sum=0; // static variables
6    sum=sum+x+y;
7    x++;
8    return sum;
9  }
10 int main()
11 {
12   // call by value
13   printf("%d\n",sum(x,y));
14   int x=10, y=20; // local variables
15   printf("%d\n",sum(x,y));
16   return 0;
17 }
```

Edit this code

ıat just executed
ine to execute

<< First | < Prev | Next > | Last >>

Step 11 of 15

ze visualization (NEW!)

Print output (drag lower right

```
3
```

Stack

Global variables

x	int	1
y	int	2

main

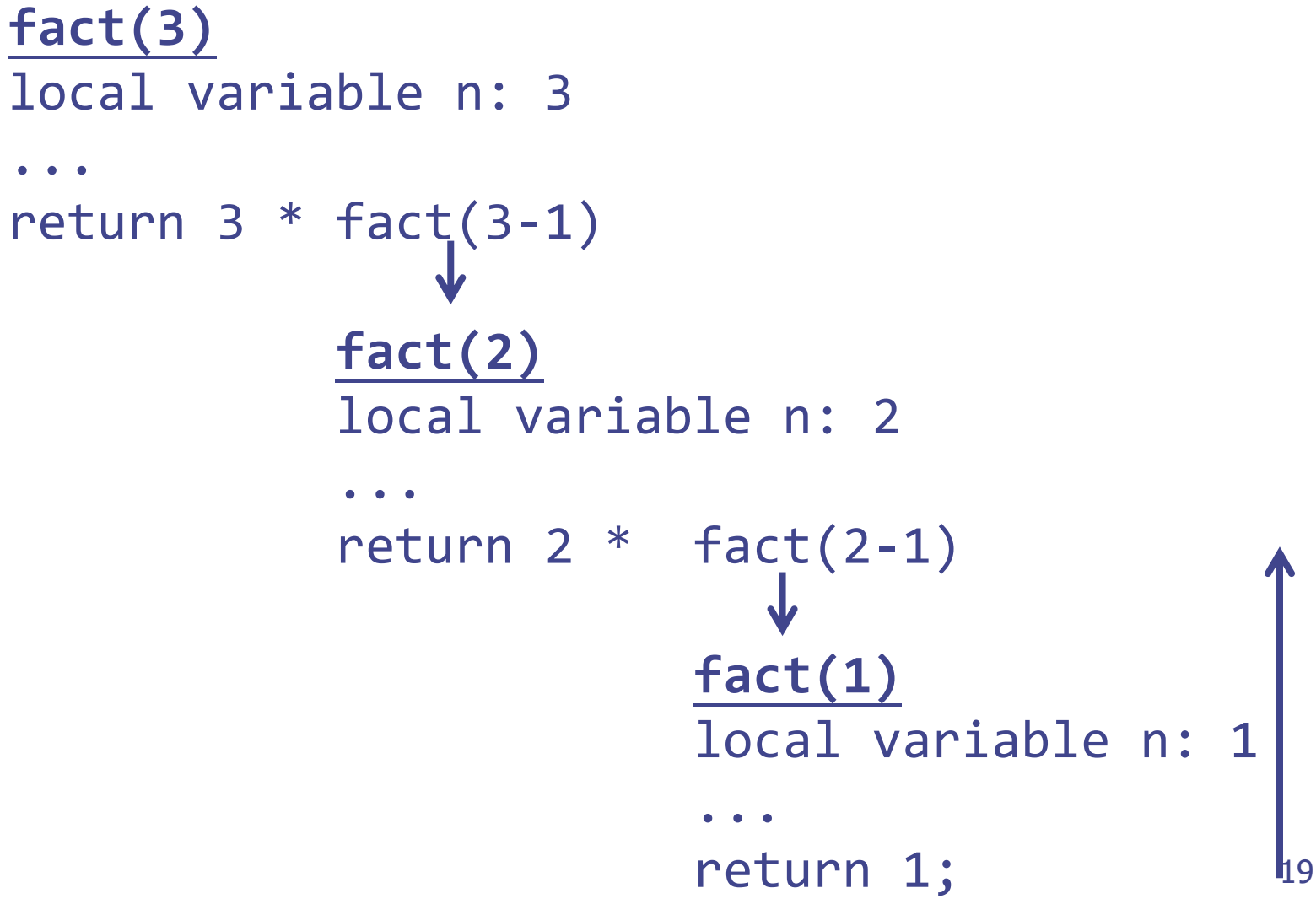
x	int	10
y	int	20

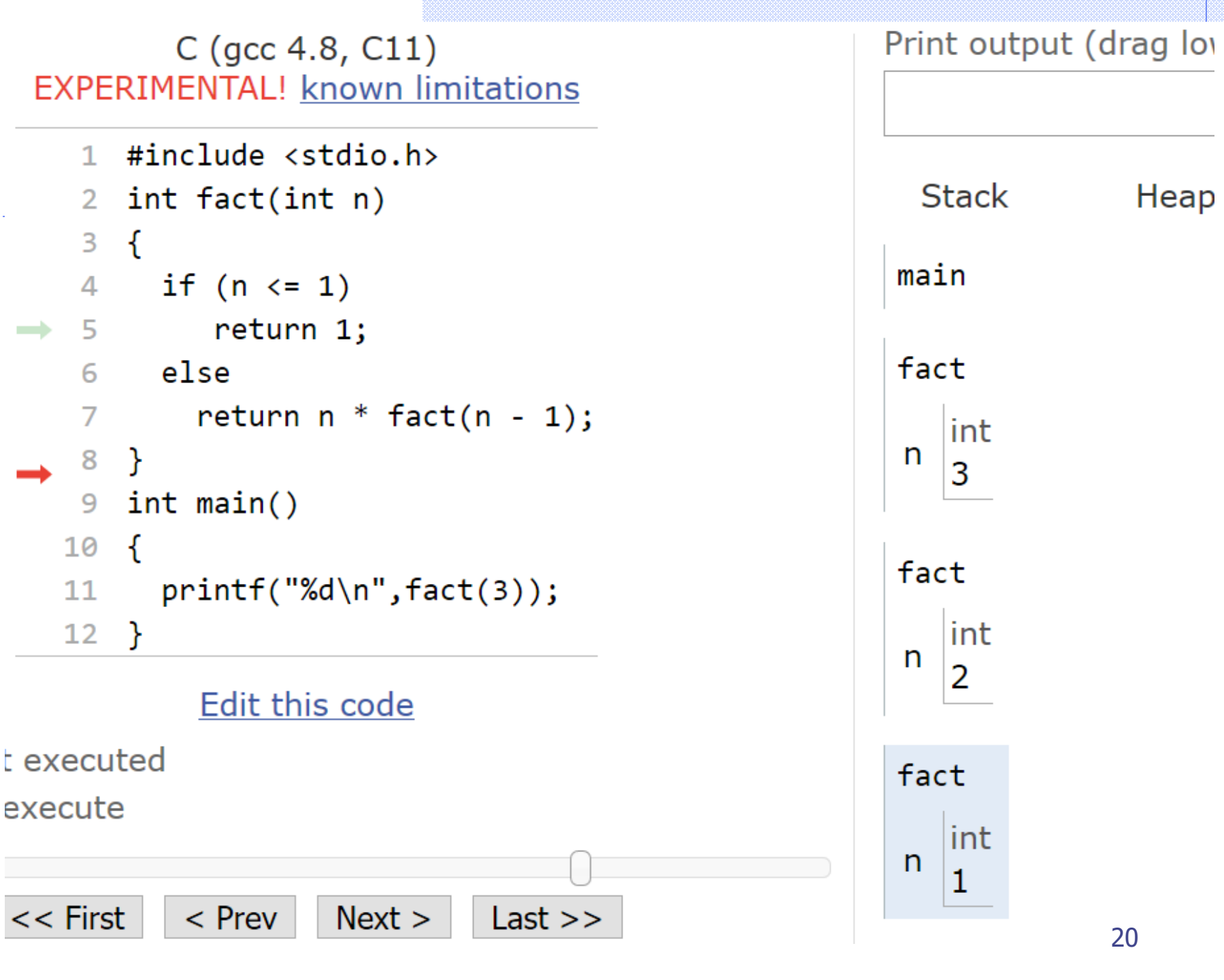
sum

x	int	10
y	int	20
sum (static 0x60104C)	int	3

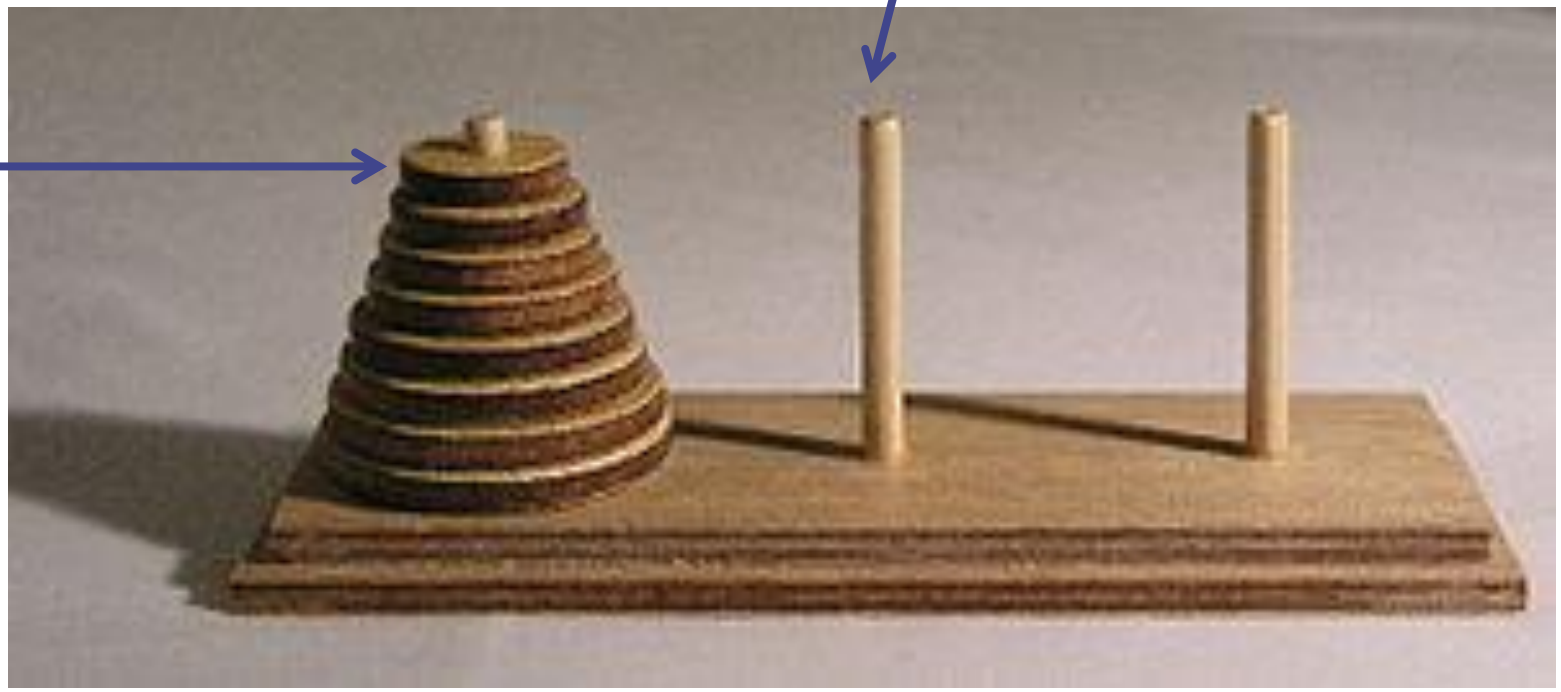
Recursion: Factorial

```
#include <stdio.h>
int fact(int n)
{
  if (n <= 1)
     return 1;
  else
    return n * fact(n - 1);
}
int main(void)
{
  printf("%d\n",fact(3));
}
```


fact(3)
local variable n: 3
...
return 3 * fact(3-1)
fact(2)
local variable n: 2
...
return 2 * fact(2-1)
fact(1)
local variable n: 1
...
return 1;


C (gcc 4.8, C11)
EXPERIMENTAL! known limitations
1 #include <stdio.h>
2 int fact(int n)
3 {
4 if (n <= 1)
5 return 1;
6 else
7 return n * fact(n - 1);
8 }
9 int main()
10 {
11 printf("%d\n",fact(3));
12 }
Edit this code
t executed
execute
<< First
< Prev
Next >
Last >>
Print output (drag lo
Stack
Heap
main
fact
n
int
3
fact
n
int
2
fact
n
int
1

Tower of Hanoi (Hanoi Tower, 河内塔)

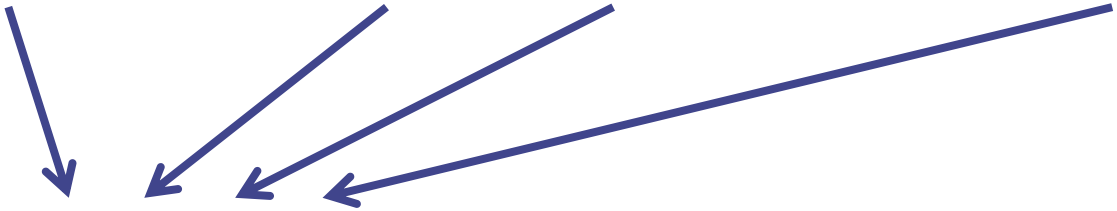


https://en.wikipedia.org/wiki/Tower_of_Hanoi

Pseudo-code

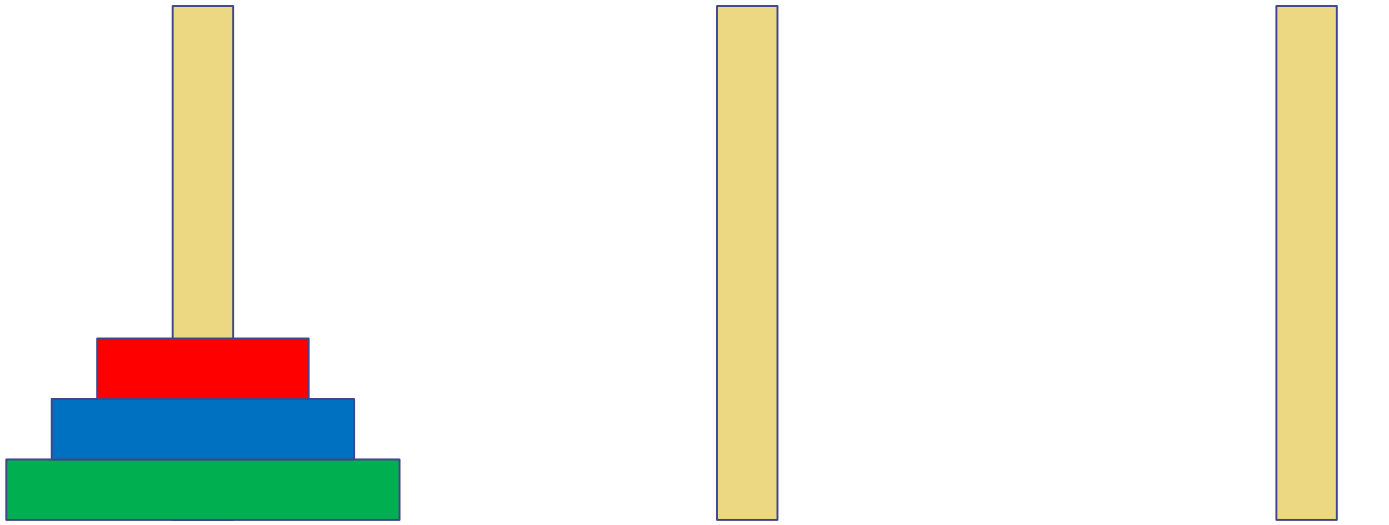
- How to move four disks from peg A to peg C?

Number of disks, Source peg, Destination peg, Buffer peg



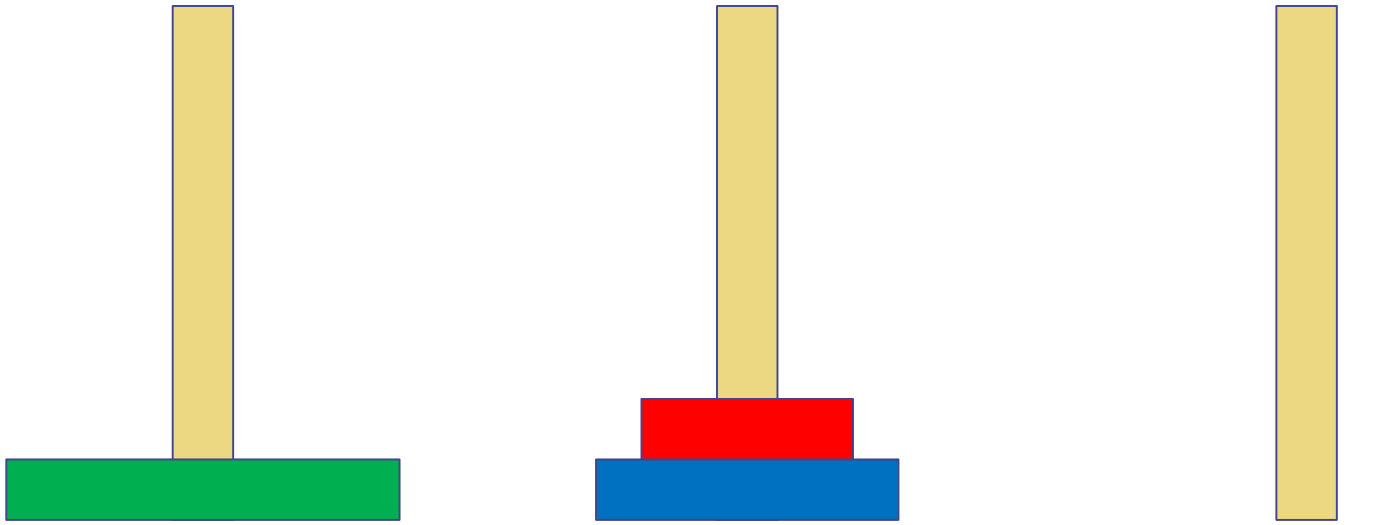
```
Move(3, A, C, B)
{
  Move(2, A, B, C)
  Move one disk from peg A to peg C
  Move(2, B, C, A)
}
```

Move(3, A, C, B)

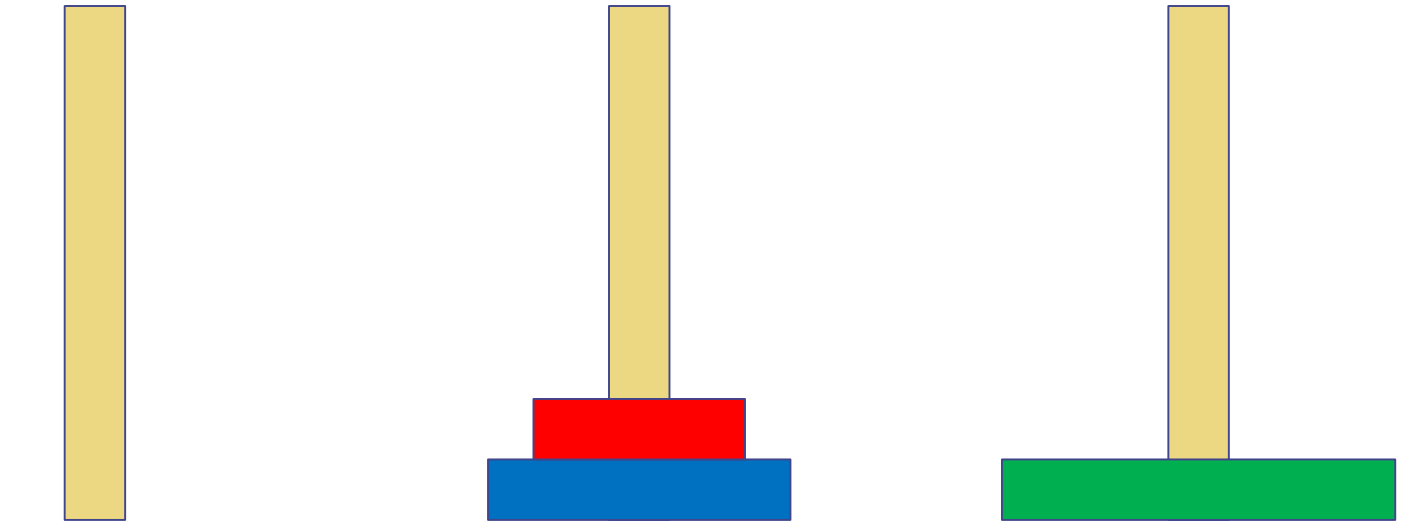


Peg A Peg B Peg C

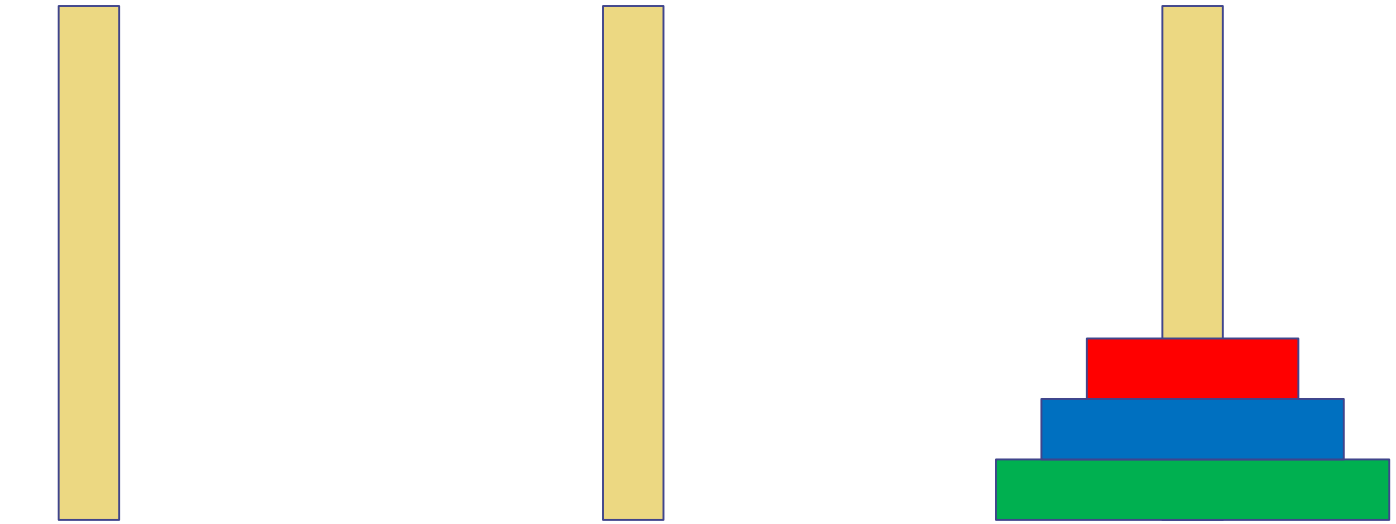
Step 1: Move(2,A,B,C)



Step 2: move one disk from peg A to peg C



Step 3: Move(2, B, C, A)



Peg A

Peg B

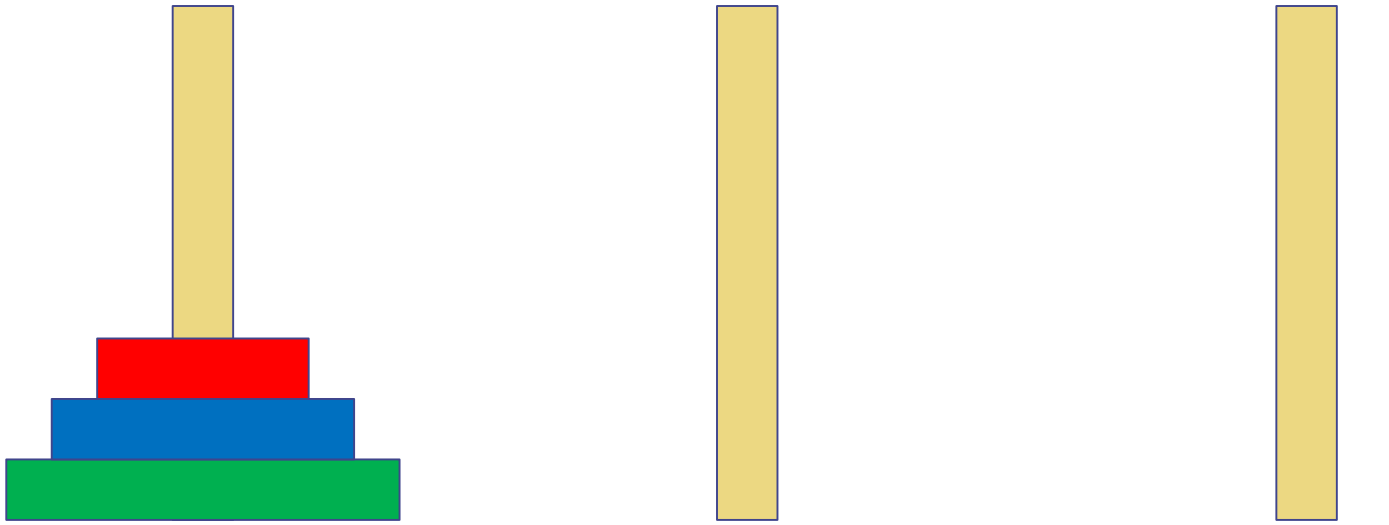
Peg C

```c
#include <stdio.h>
void Move(int no, char src, char dest, char buf)
{
  if (no<=1)
  {
    printf("Move 1 disk from peg %c to peg %c\n", src, dest);
  }
  else
  {
    Move(no-1,src,buf,dest);
    printf("Move 1 disk from peg %c to peg %c\n", src, dest);
    Move(no-1,buf,dest,src);
  }
}
int main(void)
{
  Move(3,'A','C','B');
  return 0;
}
```

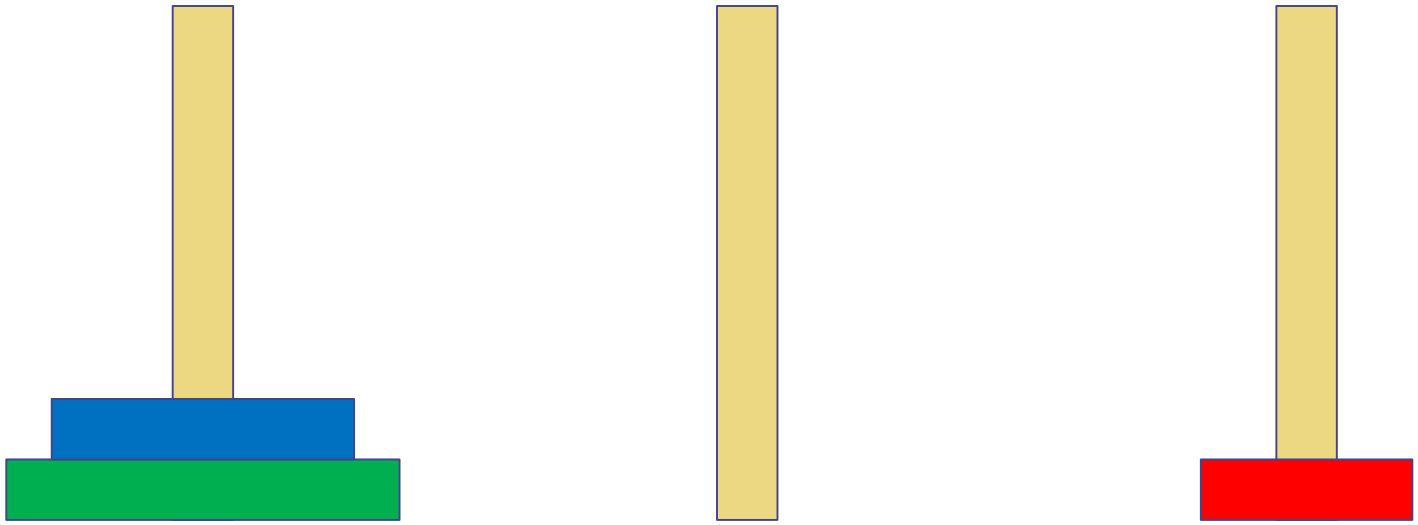
Result

- Move 1 disk from peg A to peg C
- Move 1 disk from peg A to peg B
- Move 1 disk from peg C to peg B
- Move 1 disk from peg A to peg C
- Move 1 disk from peg B to peg A
- Move 1 disk from peg B to peg C
- Move 1 disk from peg A to peg C

Move(3, A, C, B)



Move 1

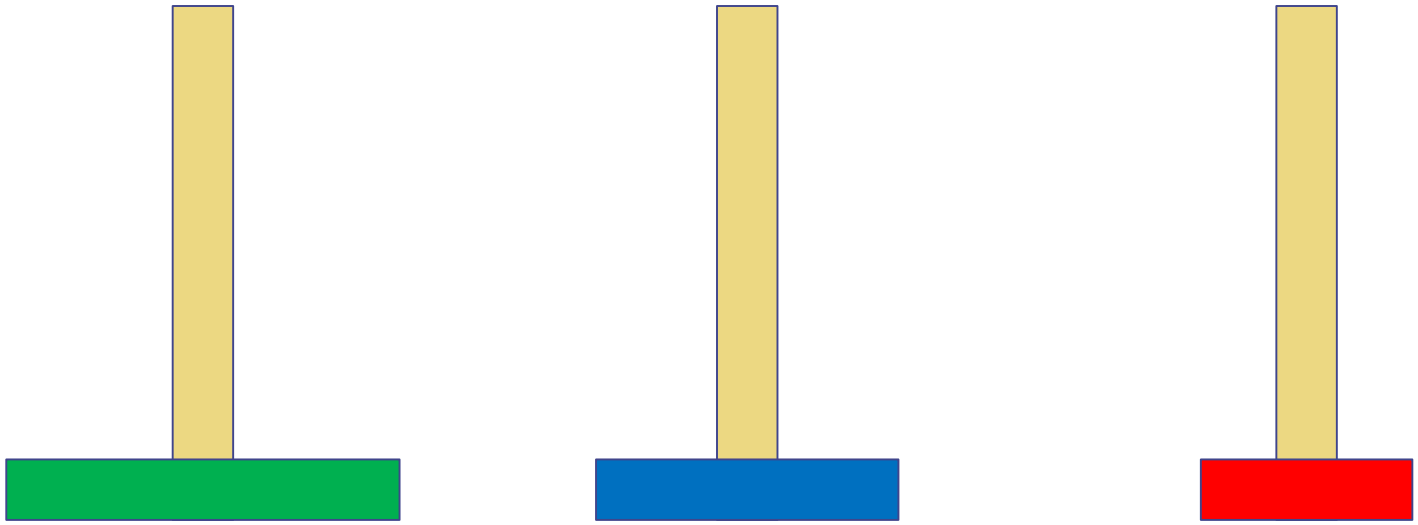


Peg A

Peg B

Peg C

Move 2

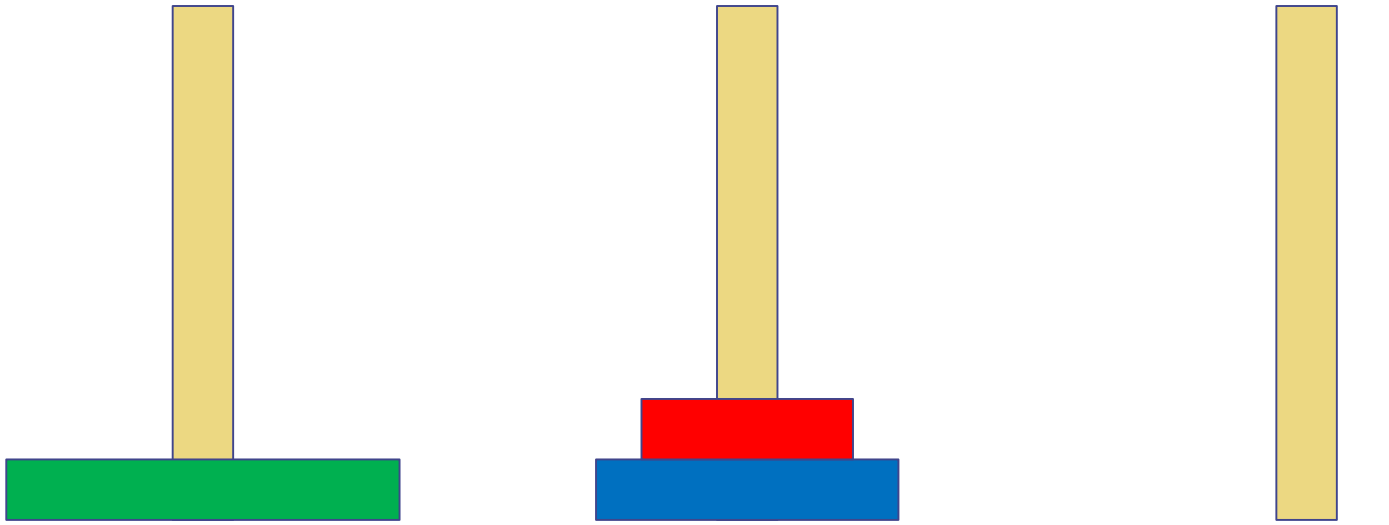


Peg A

Peg B

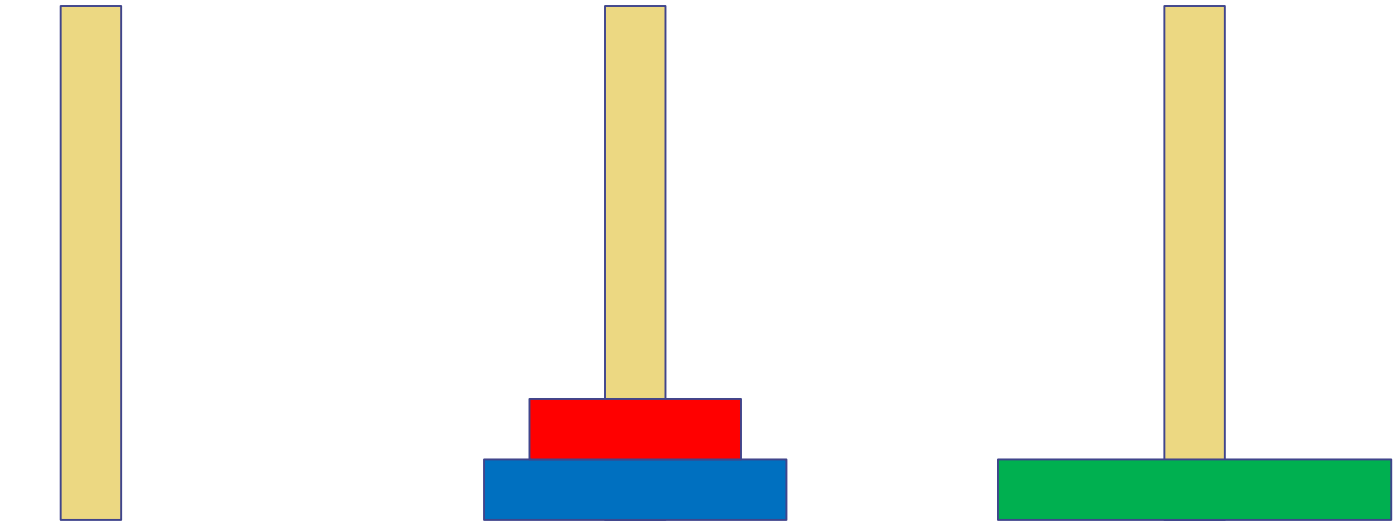
Peg C

Move 3



Peg A Peg B Peg C

Move 4

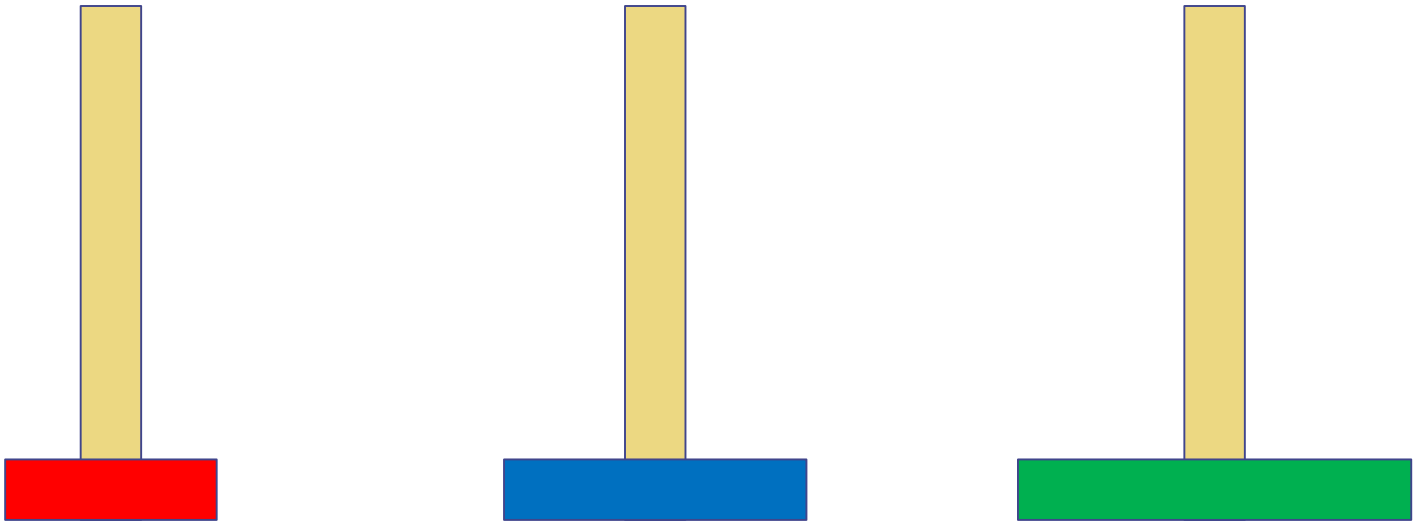


Peg A

Peg B

Peg C

Move 5

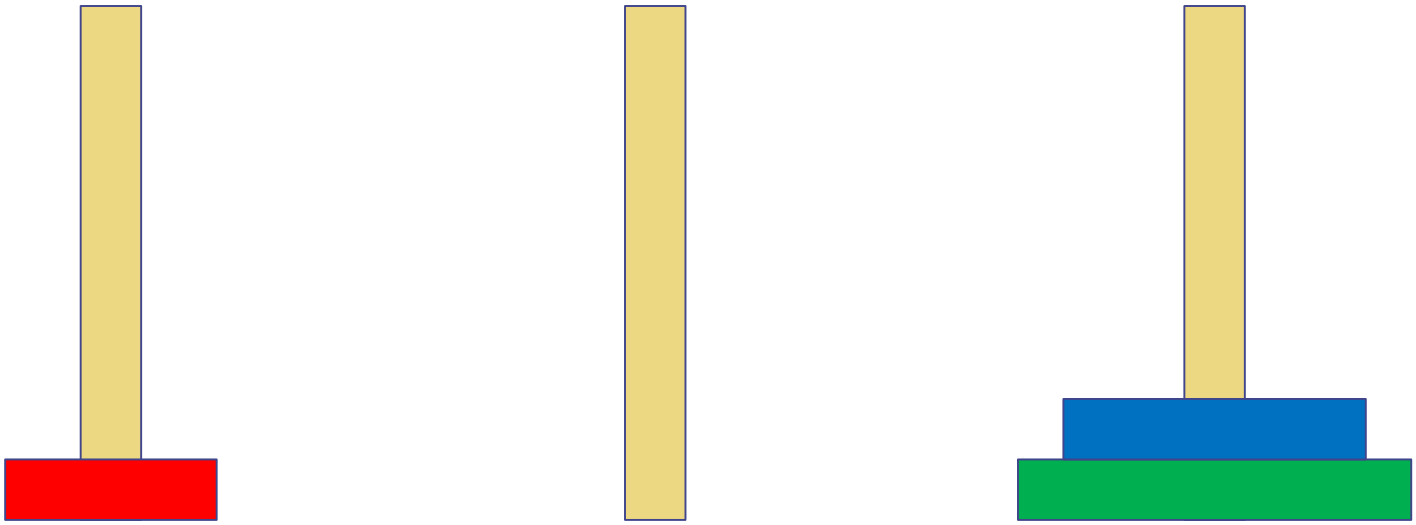


Peg A

Peg B

Peg C

Move 6

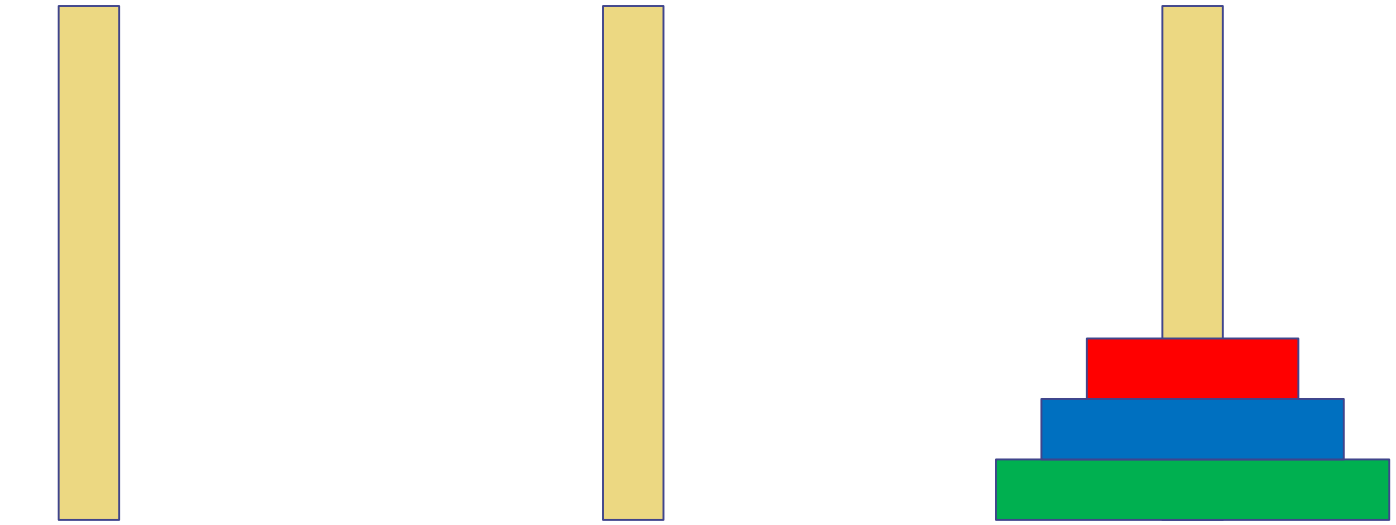


Peg A

Peg B

Peg C

Move 7



Peg A

Peg B

Peg C

Practice

- Sum: 1+2+3…+n
  - sum(n)=sum(n-1)+n, when n>1
  - sum(1)=1
- Fibonacci sequence (費氏數列)
  - fib(n)=fib(n-1)+fib(n-2) when n>2
  - fib(1)=1
  - fib(2)=1

Sum: Recursion Version

```
#include <stdio.h>
int sum(int n)
{
  if (n<=1)
    return 1;
  else
    return n+sum(n-1);
}
int main(void)
{
  printf("%d", sum(10));
  return 0;
}
```

Sum: Iteration/Loop Version

```
#include <stdio.h>
int sum(int n)
{
  int i,sum;
  for(i=1,sum=0;i<=n;i++)
    sum+=i;
  return sum;
}
int main(void)
{
  printf("%d", sum(10));
  return 0;
}
```

Sum: Equation Version

```
#include <stdio.h>
int sum(int n)
{
  if (n<=1)
    return 1;
  else
    return (1+n)*n/2;
}
int main(void)
{
  printf("%d", sum(10));
  return 0;
}
```

Fibonacci Sequence: Recursion Version

```
#include <stdio.h>
int fib(int n)
{
  if (n<=2)
    return 1;
  else
    return fib(n-2)+fib(n-1);
}
int main(void)
{
  printf("%d", fib(9));
  return 0;
}
```

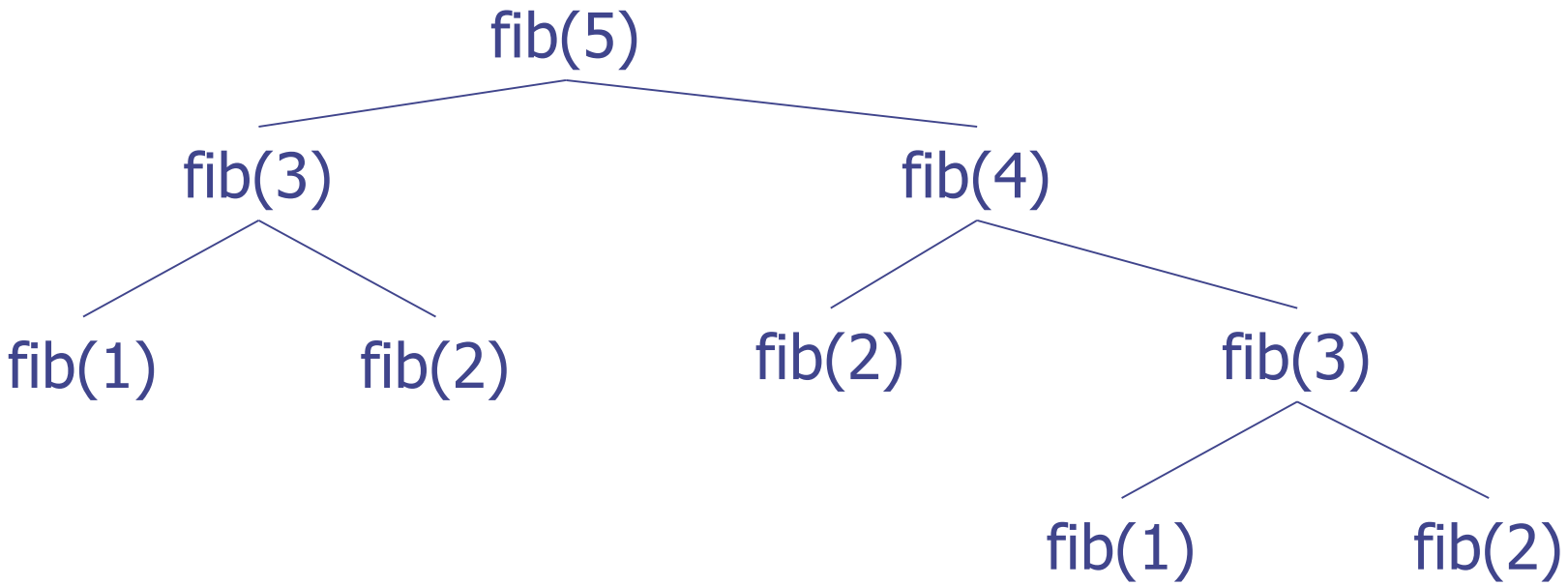
fib() is called 34 times

Fibonacci Sequence: Recursion Version With Table

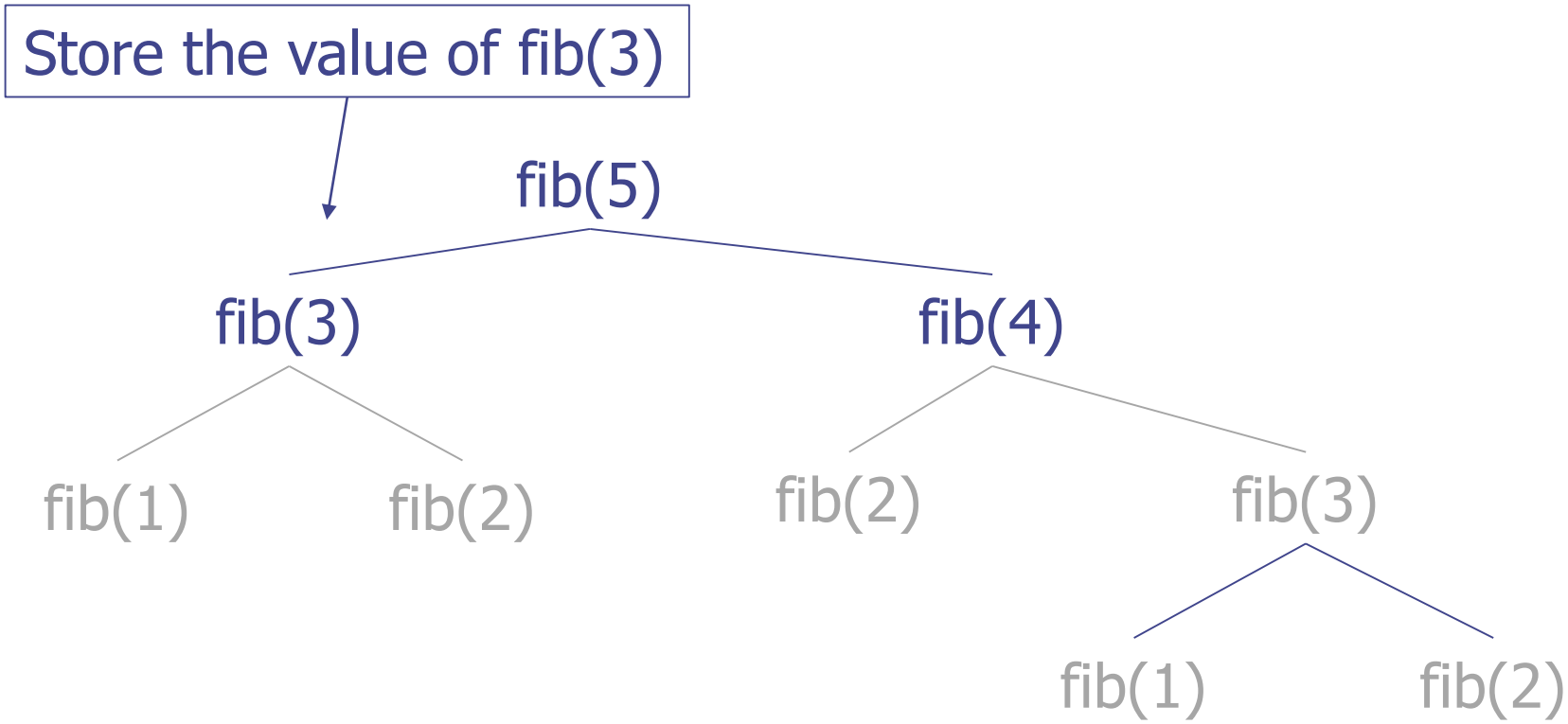
```
int fib_number[10]={0,1,1};
int fib(int n)
{
  int f_2, f_1;
  if (n<=2)
    return 1;
  else
  {
    f_2=fib_number[n-2]>0 ? fib_number[n-2] : fib(n-2);
    f_1=fib_number[n-1]>0 ? fib_number[n-1] : fib(n-1);
    fib_number[n]=f_2+f_1;
    return fib_number[n];
  }
}
```

fib() is called 7 times

Calculating fib(5) Without Table



Calculating fib(5) With Table



Fibonacci Sequence: Iteration/Loop Version

```
int fib(int n)
{
  int f_2=0, f_1=1, f=1;
  if (n<=2)
    return 1;
  else {
    for(int i=3;i<=n;i++) {
      f_2=f_1;
      f_1=f;
      f=f_2+f_1;
    }
    return f;
  }
}
```