UCSC Silicon Valley Extension

Advanced C Programming

Recursion

Instructor: Radhika Grover

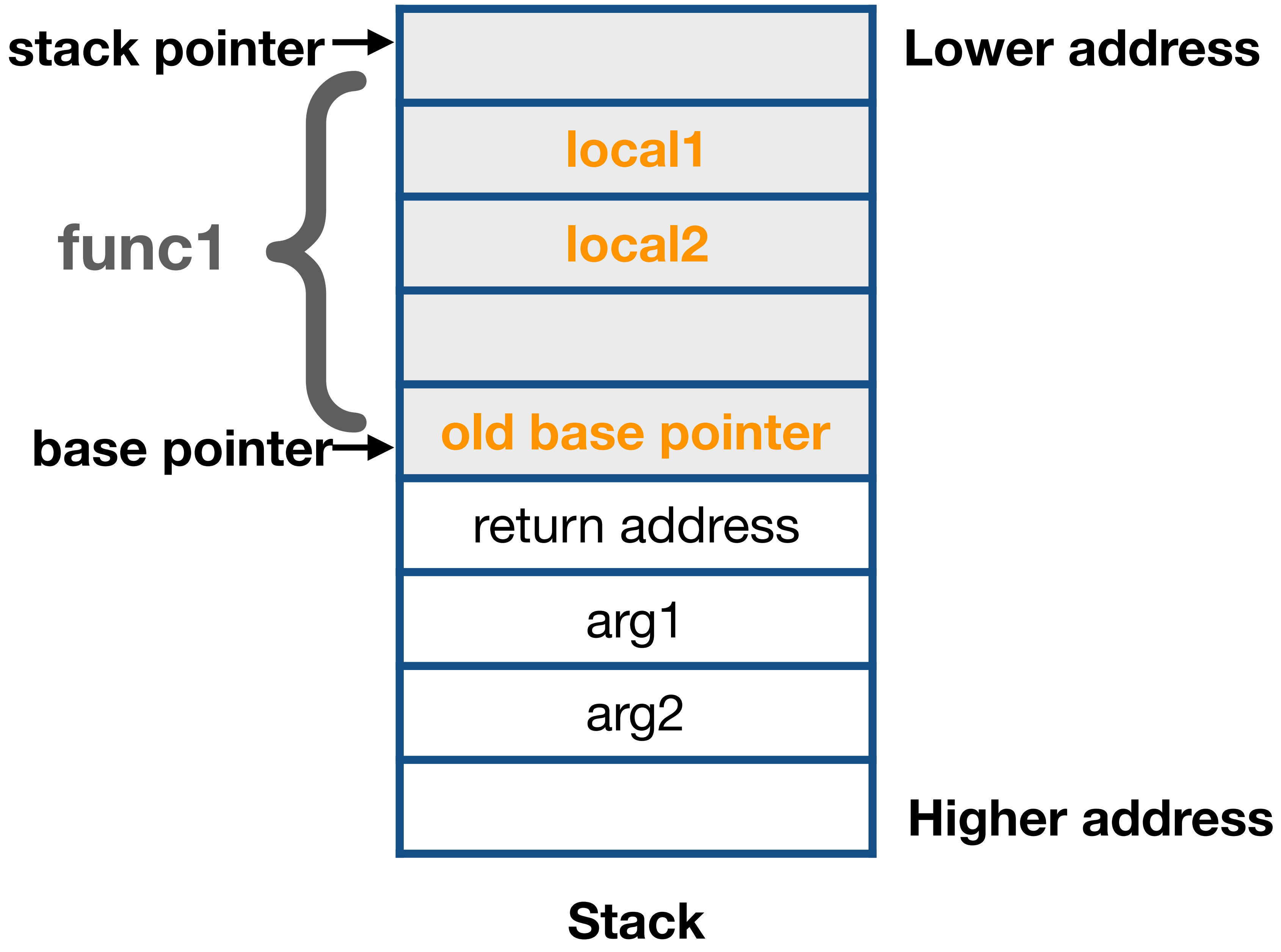
Overview

- Call stack
- Call chain
- Examples using recursion

Call stack

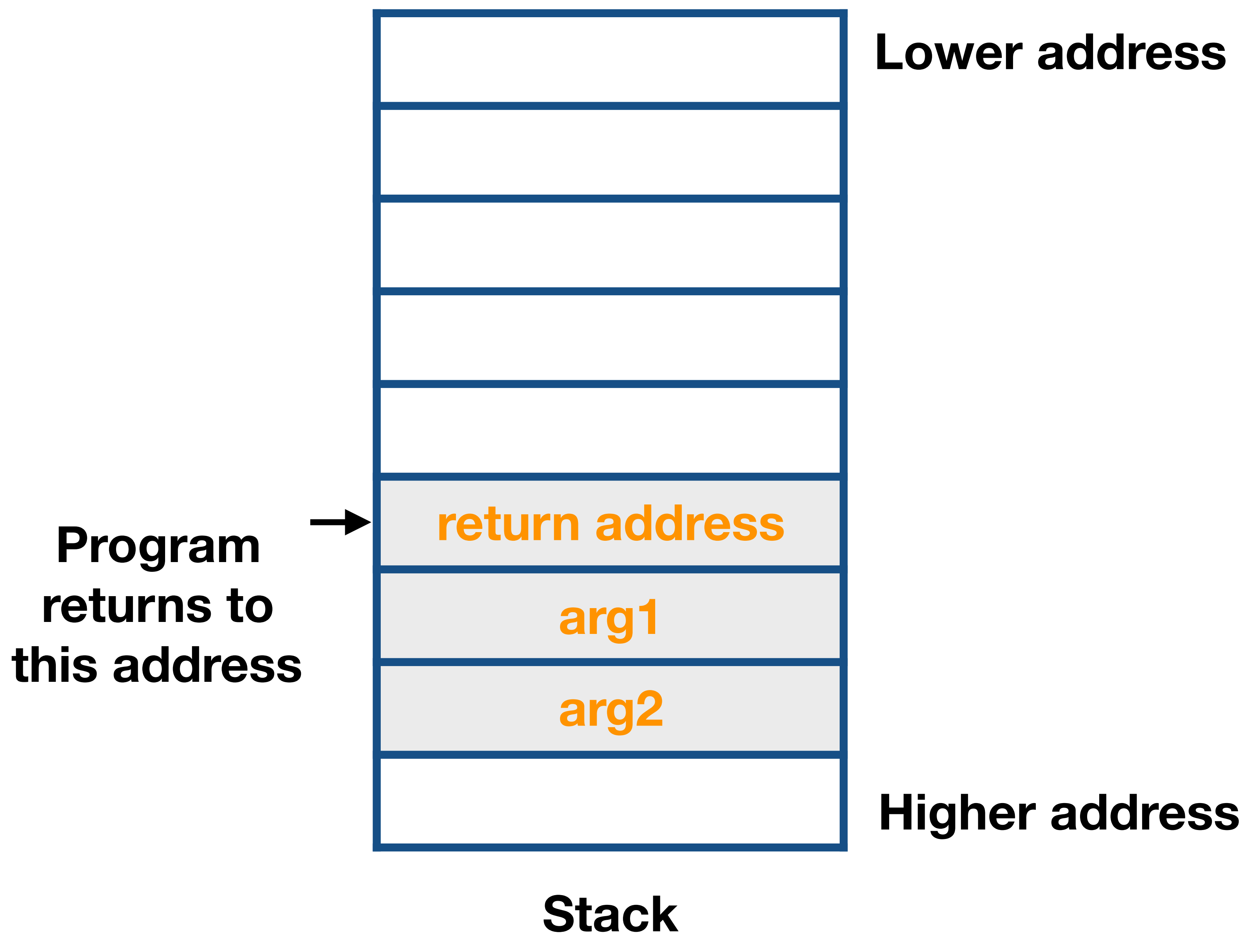
- Every function that is executing stores its data on the stack

```
int func1(int
param1, int param2){
  int local1, local2;
  // some code
}
func1(arg1, arg2);
```



Call stack

```
int func1(int
param1, int param2){
  int local1,
  local2;
  // some code
  return data;
}
```

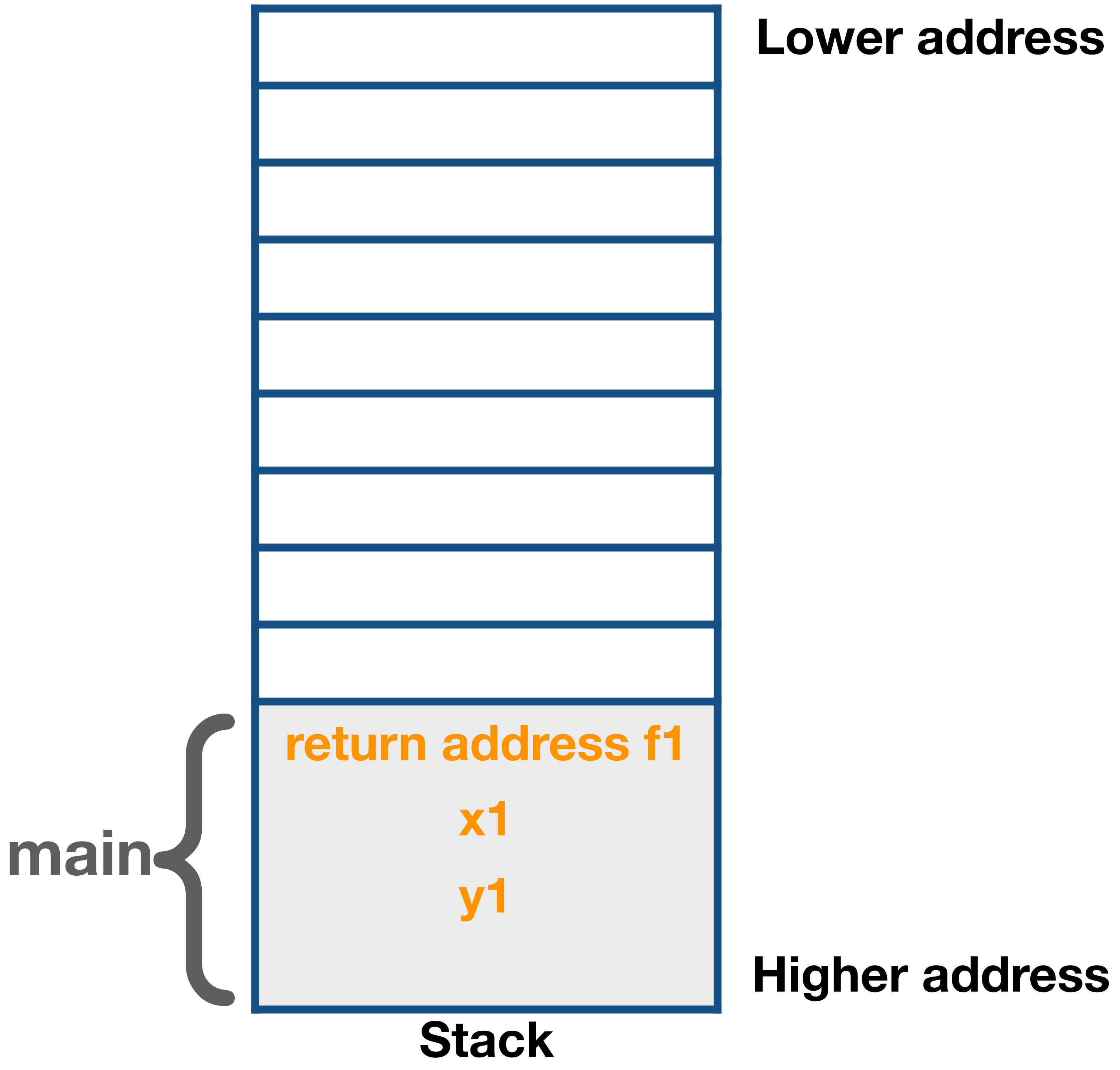


Call chain

main( )

```
int main(){
  f1(x1, y1);
}
```

← return address f1

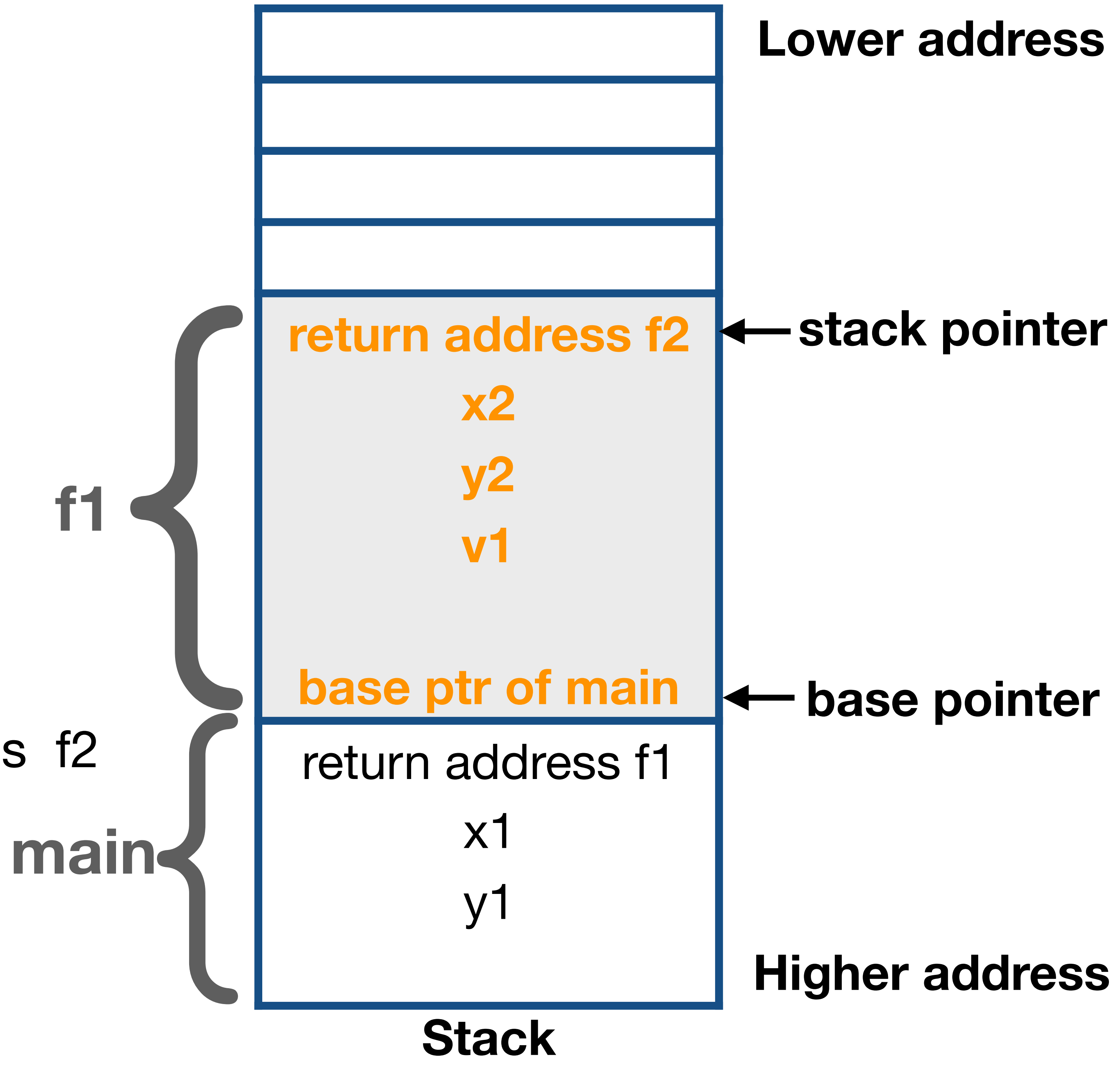


Call chain

main( ) -> f1(x1,y1)

```
int main(){
  f1(x1, y1);
}

int f1(int param1,
int param2){
  …
  v1 = f2(x2, y2);
  return v1;
}
```

← return address f2

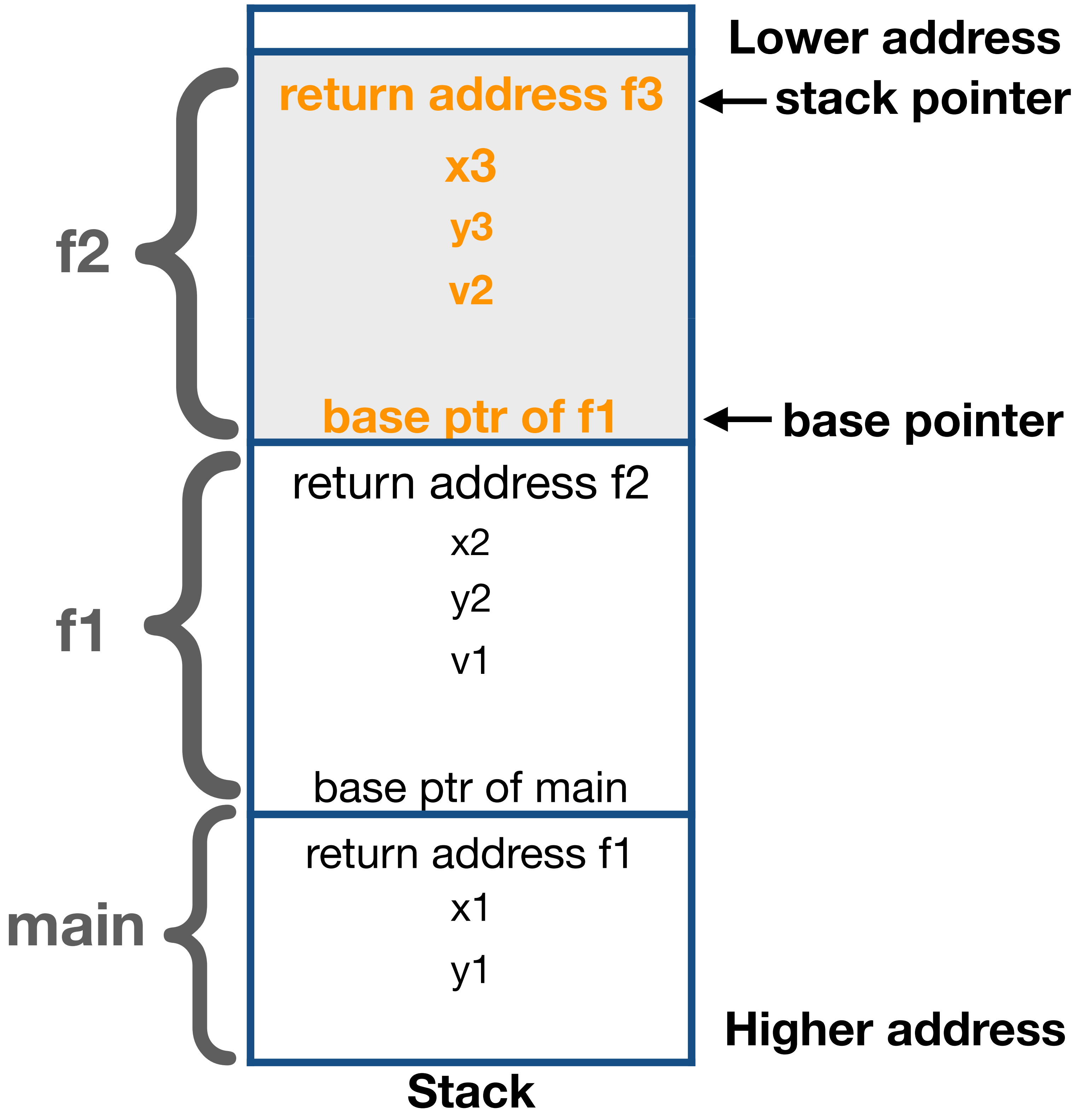


Call chain

main( ) -> f1(x1, y1) -> f2(x2, y2)

```
int f1(int param1, int
param2){
  …
  v1 = f2(x2, y2);
  return v1;
}

int f2(int param1, int
param2){
  …
  v2 = f3(x3, y3);
  return v2;
}
```

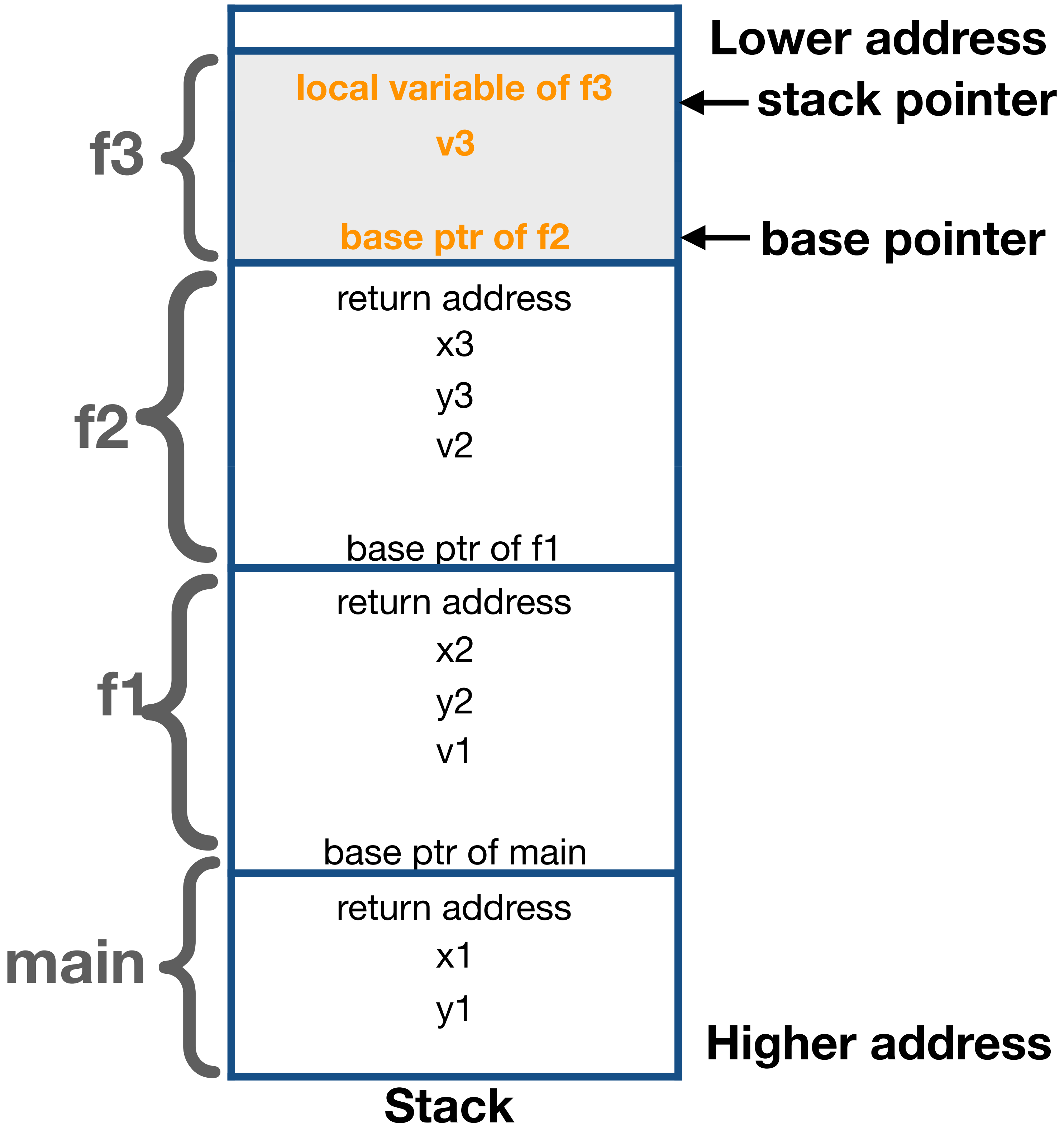
← return address f3



Call chain

main( ) -> f1(x1, y1) -> f2(x2, y2) -> f3(x3, y3)

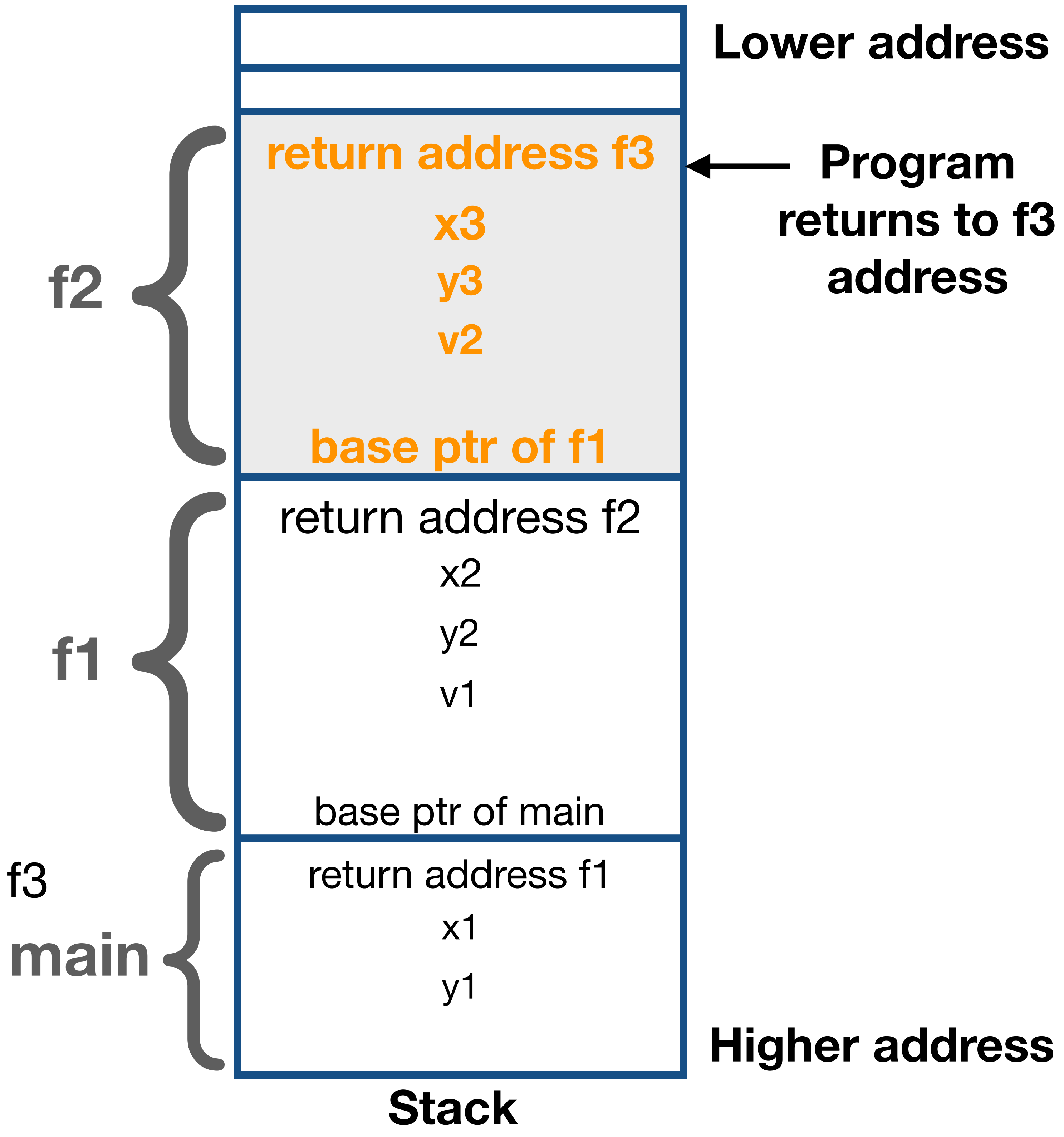
```
int f2(int param1, int
param2){
  v2 = f3(x3, y3);
  return v2;
}

int f3(int param1, int
param2){
  return v3;
}
```



Call chain

f3 executes return statement

```
int f3(int param1, int
param2){
  return v3;
}

int f2(int param1, int
param2){
  v2 = f3(x3, y3); ← return address  f3
  return v2;
}
```



Lower address

return address f3

x3

y3

v2

base ptr of f1

f2

Program returns to f3 address

return address f2

x2

y2

v1

base ptr of main

f1

return address f1

x1

y1

main

Higher address

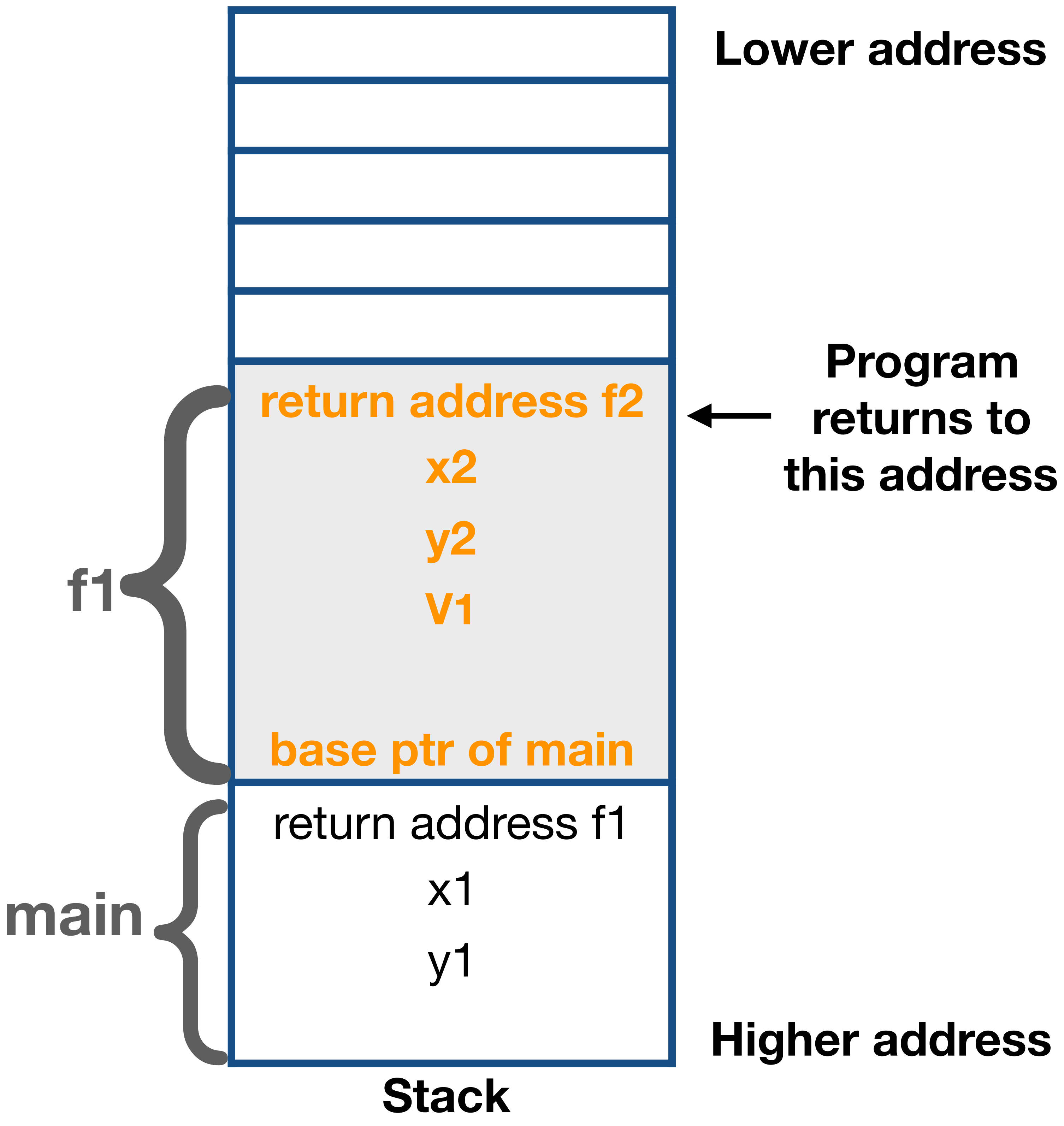
Stack

Call chain

f2 executes return statement

```
int f2(int param1, int
param2){
  v2 = f3(x3, y3);
  return v2;
}

int f1(int param1, int
param2){
  v1 = f2(x2, y2);
  return v1;
}
```

← return address f2



Lower address

return address f2

x2

y2

V1

base ptr of main

return address f1

x1

y1

f1

main

Program returns to this address

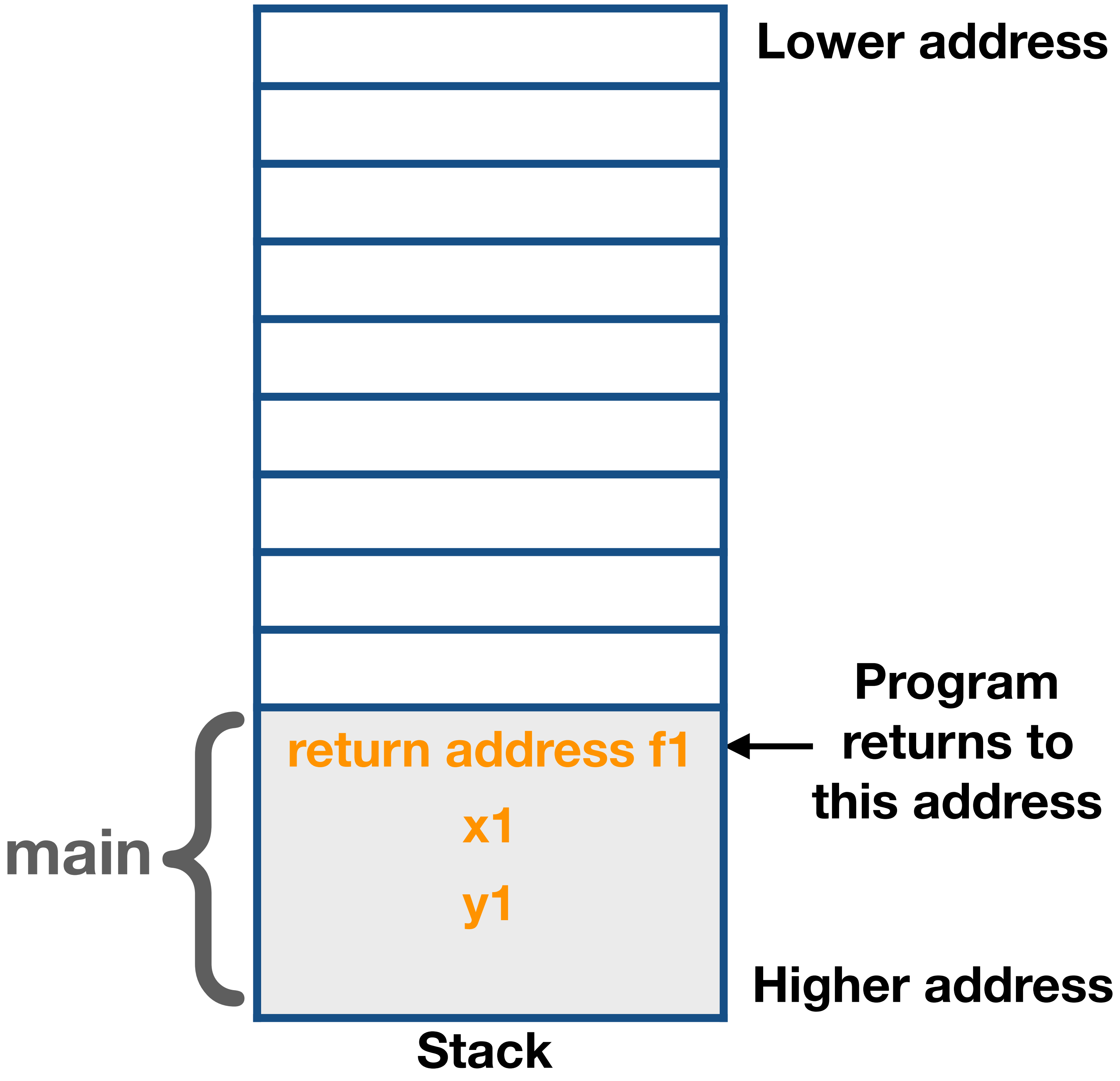
Higher address

Stack

Call chain

f1 executes return statement

```
int f1(int param1,
int param2){
  v1 = f2(x2, y2);
  return v1;
}

int main(){
  f1(x1, y1); ← return address  f1
}
```



Recursive function

Exercise : show the call stack of the following recursive function

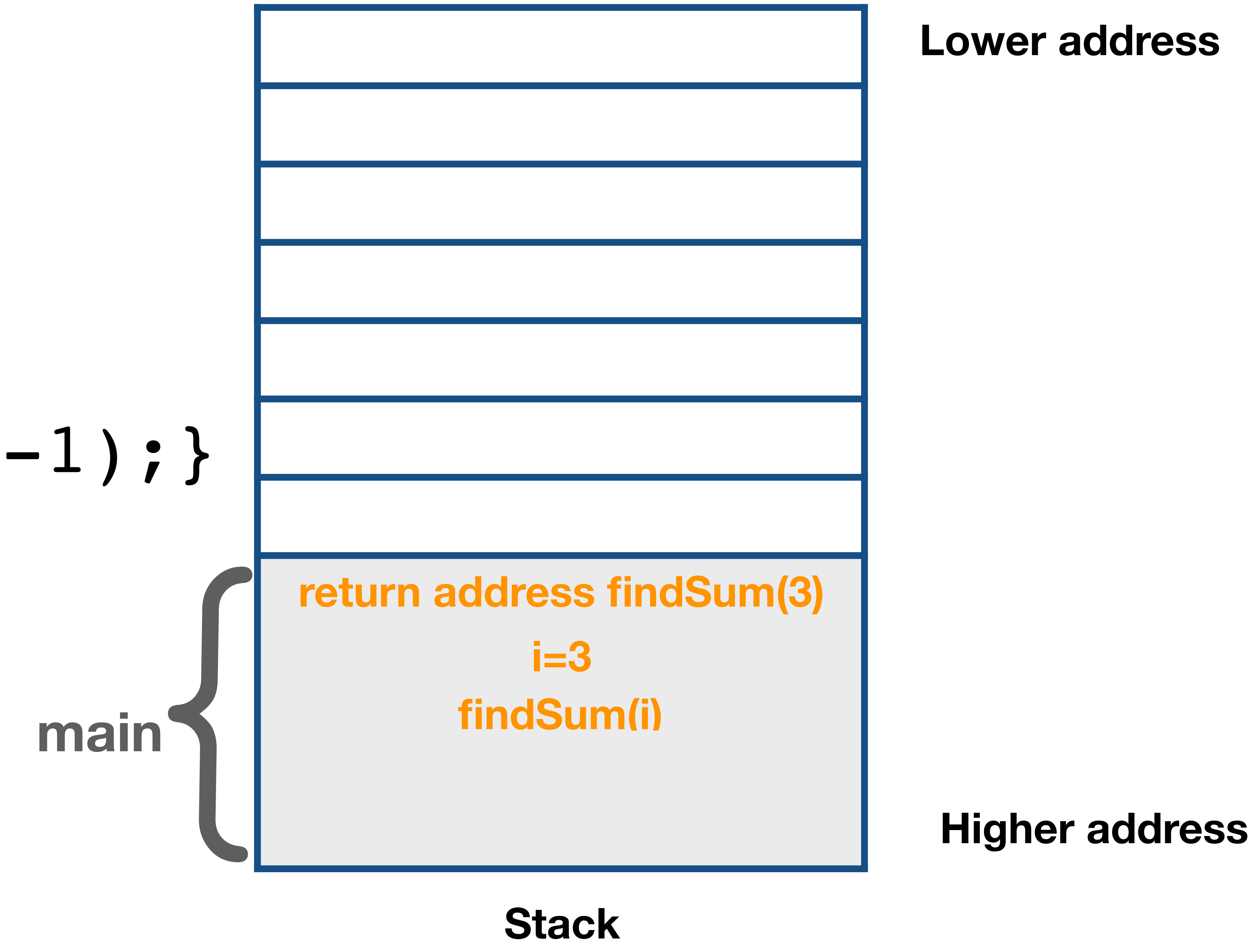
```
void displaySquare(int i) {
  cout << i*i << endl;
  if (i > 0)
    displaySquare(i-1); }

int main() {
  int i = 10;
  displaySquare(i); }
```

Solution

Recursive function

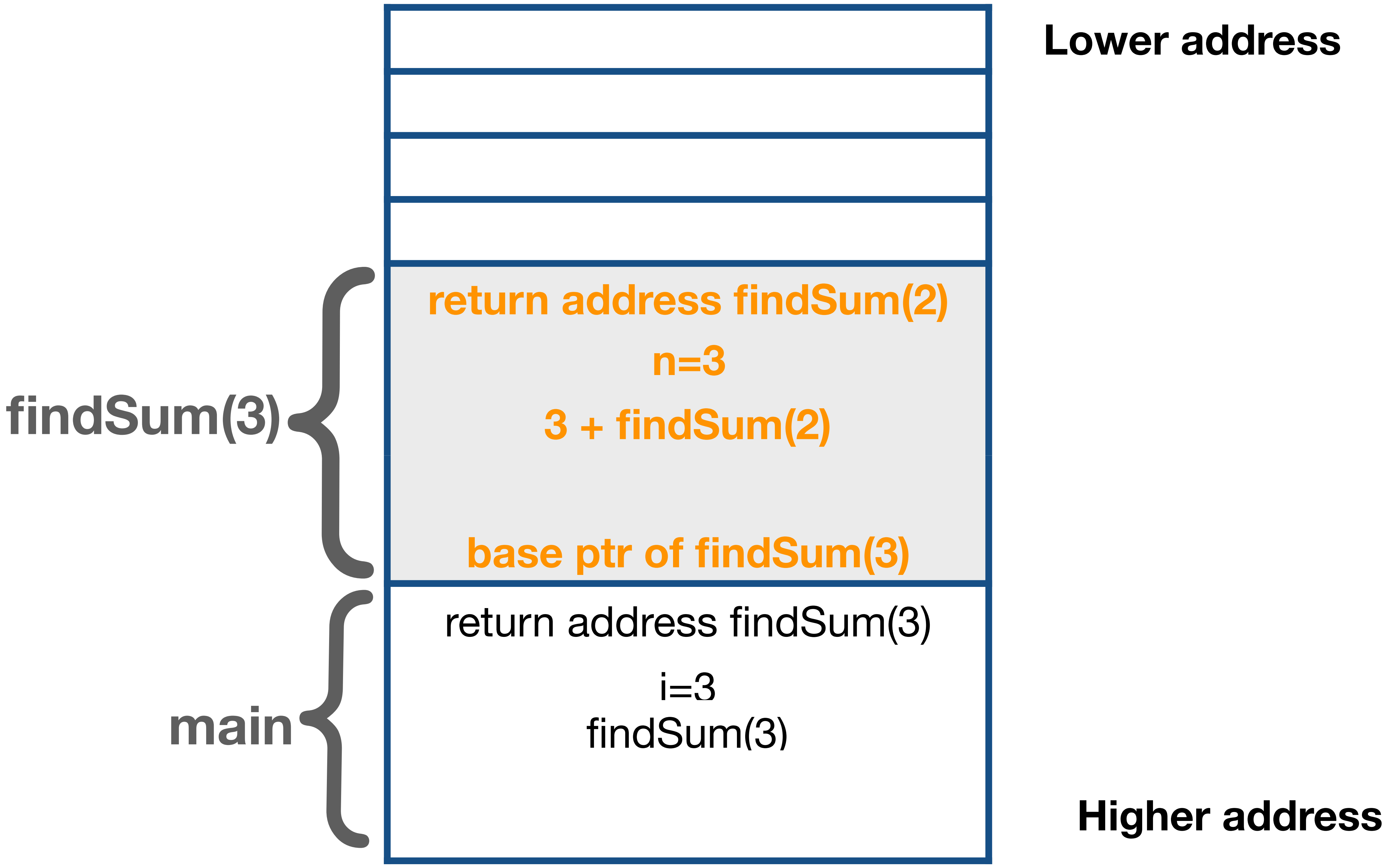
Write a recursive function that finds the sum of n numbers.

```
 int findSum(int n){
   if(n == 1)
    return 1;
   else
    return n + findSum(n-1);}

int main() {
  int i = 3;
  findSum(i); }
```



Recursive function

```
n = 3
int findSum(n){
 if(n == 1)
  return 1;
 else
  return 3 +
  findSum(2);
}
```



Lower address

findSum(3)

return address findSum(2)

n=3

3 + findSum(2)

base ptr of findSum(3)

main

return address findSum(3)

i=3

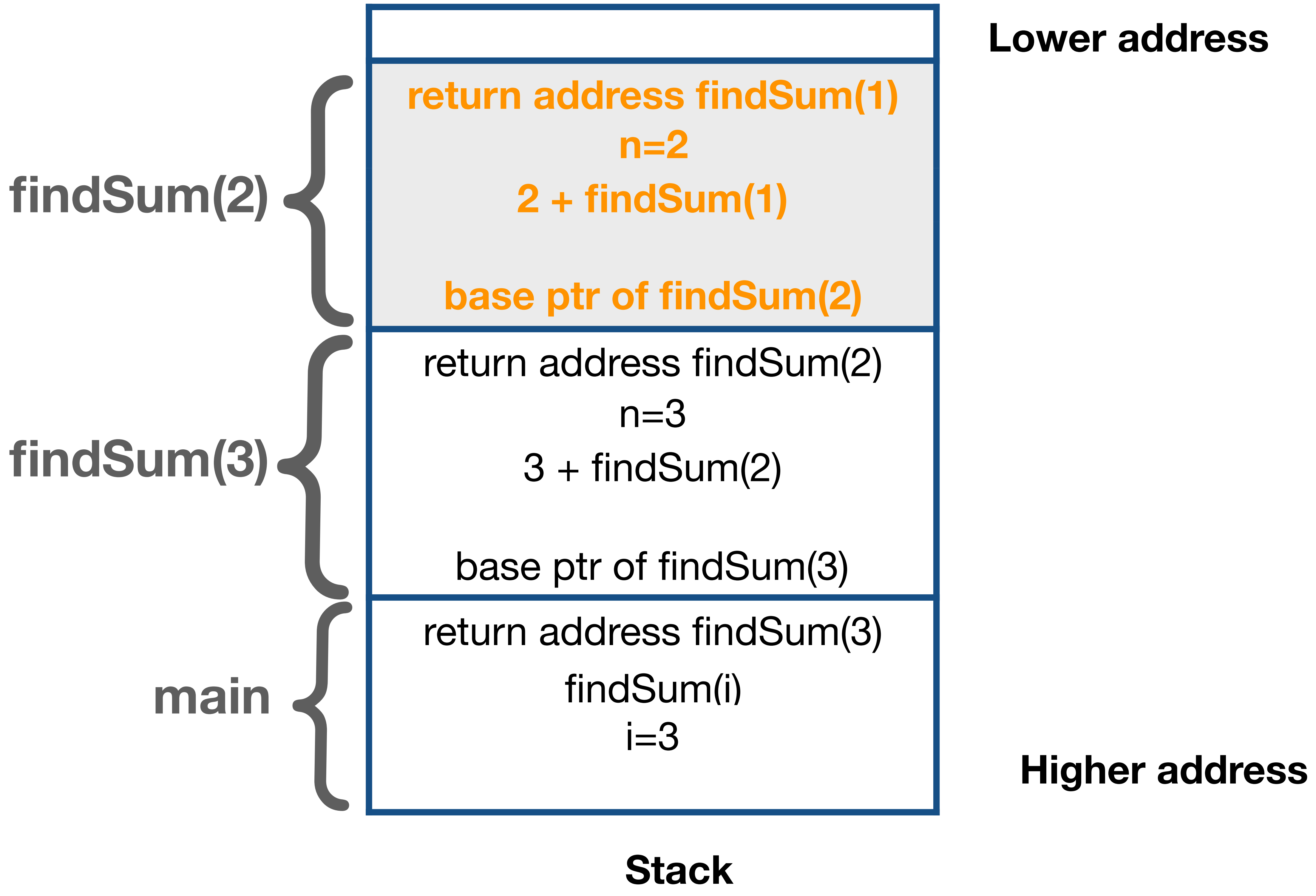
findSum(3)

Higher address

Stack

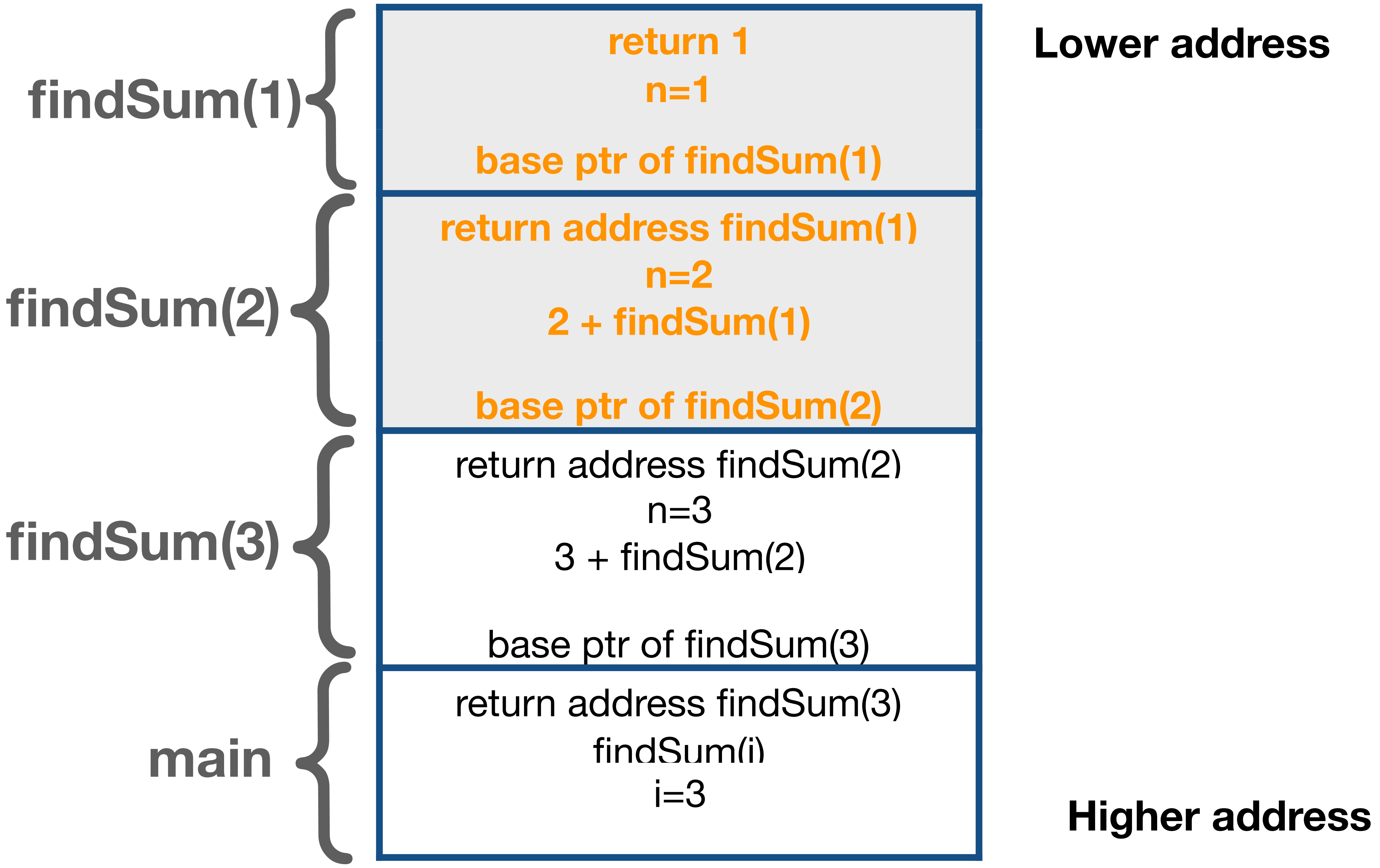
Recursive function

```
 n = 2
 int findSum(n){
   if(n == 1)
    return 1;
   else
    return 2 +
    findSum(1);
}
```



Recursive function

```
n = 1
int findSum(n){
  if(n == 1)
   return 1;
  else
   return 2 +
   findSum(1);
}
```



findSum(1)

return 1
n=1

base ptr of findSum(1)

findSum(2)

return address findSum(1)
n=2
2 + findSum(1)

base ptr of findSum(2)

findSum(3)

return address findSum(2)
n=3
3 + findSum(2)

base ptr of findSum(3)

main

return address findSum(3)
findSum(i)
i=3

Lower address

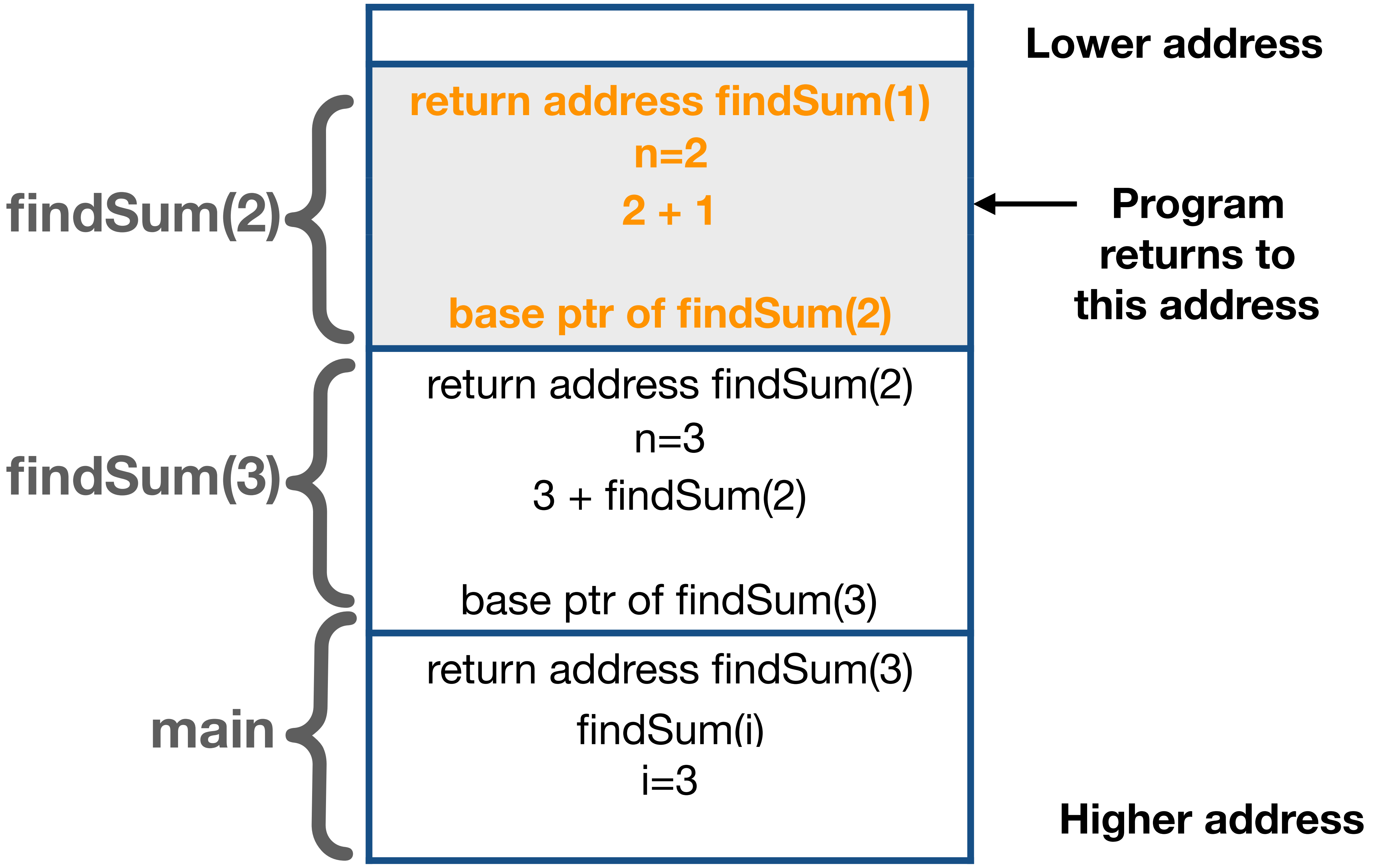
Higher address

Stack

Recursive function

```
int findSum(n){
  if(n == 1)
   return 1;
  else
   return 2 +
   findSum(1);
}
```

↑ return findSum(1)



findSum(2)

return address findSum(1)
n=2
2 + 1

base ptr of findSum(2)

findSum(3)

return address findSum(2)
n=3
3 + findSum(2)

base ptr of findSum(3)

main

return address findSum(3)
findSum(i)
i=3

Lower address

Program returns to this address

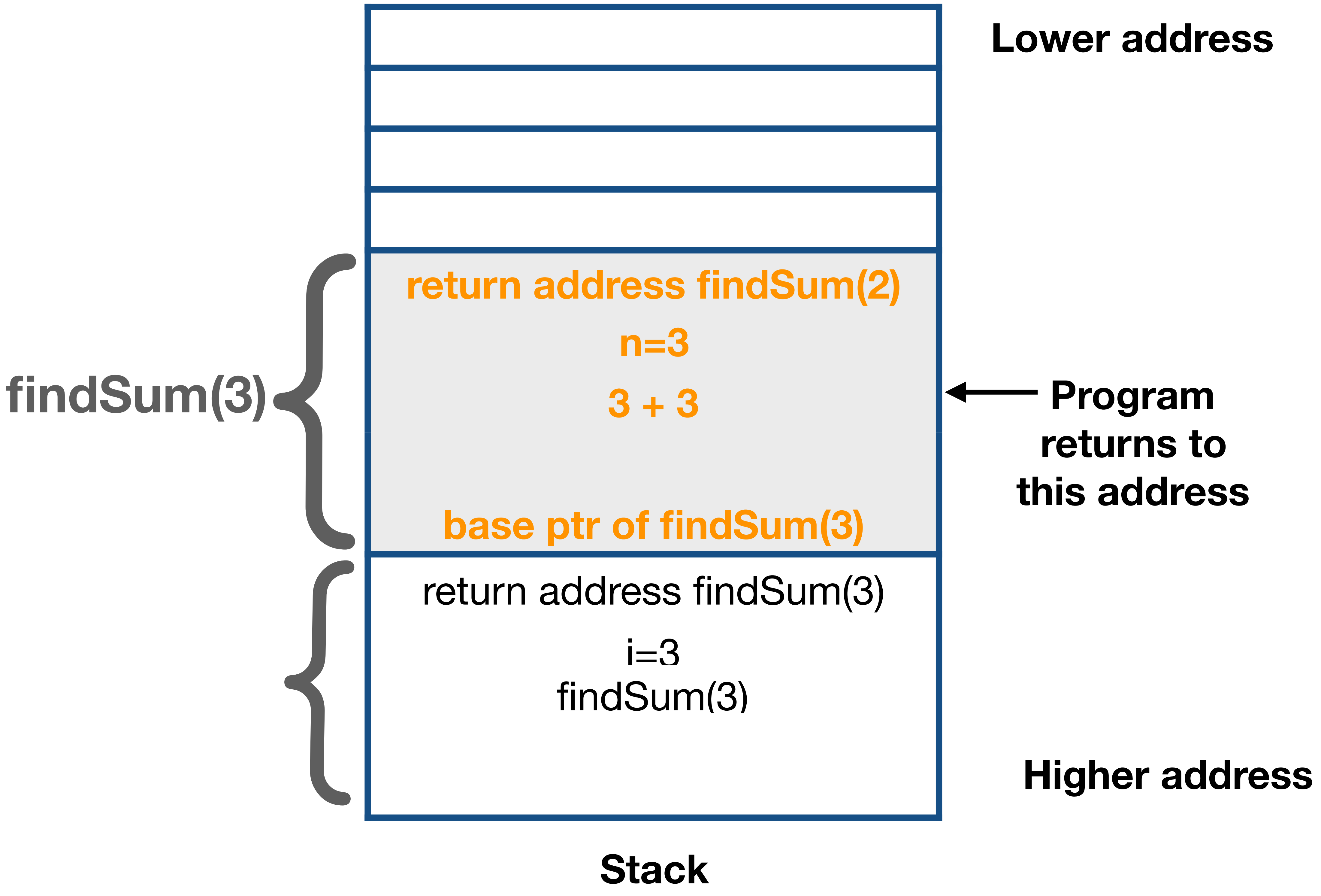
Higher address

Stack

Recursive function

```
int findSum(n){
  if(n == 1)
   return 1;
  else
   return 3 +
   findSum(2);
}
```

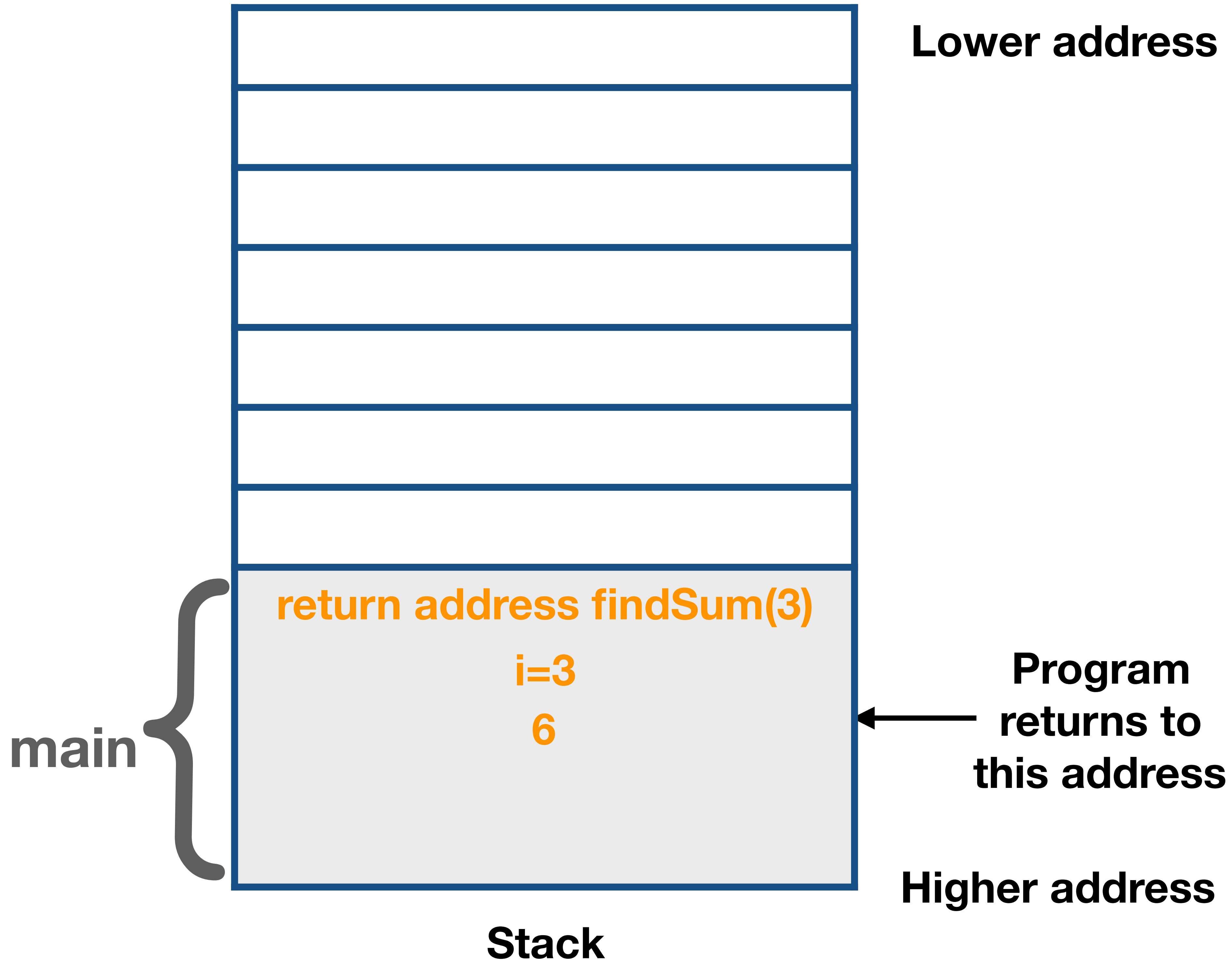
return findSum(2)



Recursive function

```
int findSum(int n){
  if(n == 1)
   return 1;
  else
   return n +
   findSum(n-1);}

int main() {
  int i = 3;
  findSum(i);
}
```

return findSum(3)



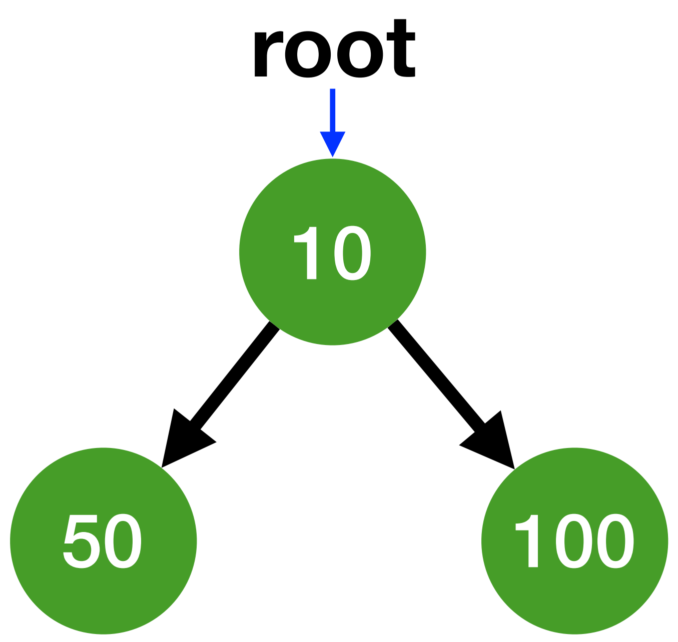
Recursive function

What is the result if find sum modified as follows ?

```
int findSum(int n){
  return n + findSum(n-1);
}
```

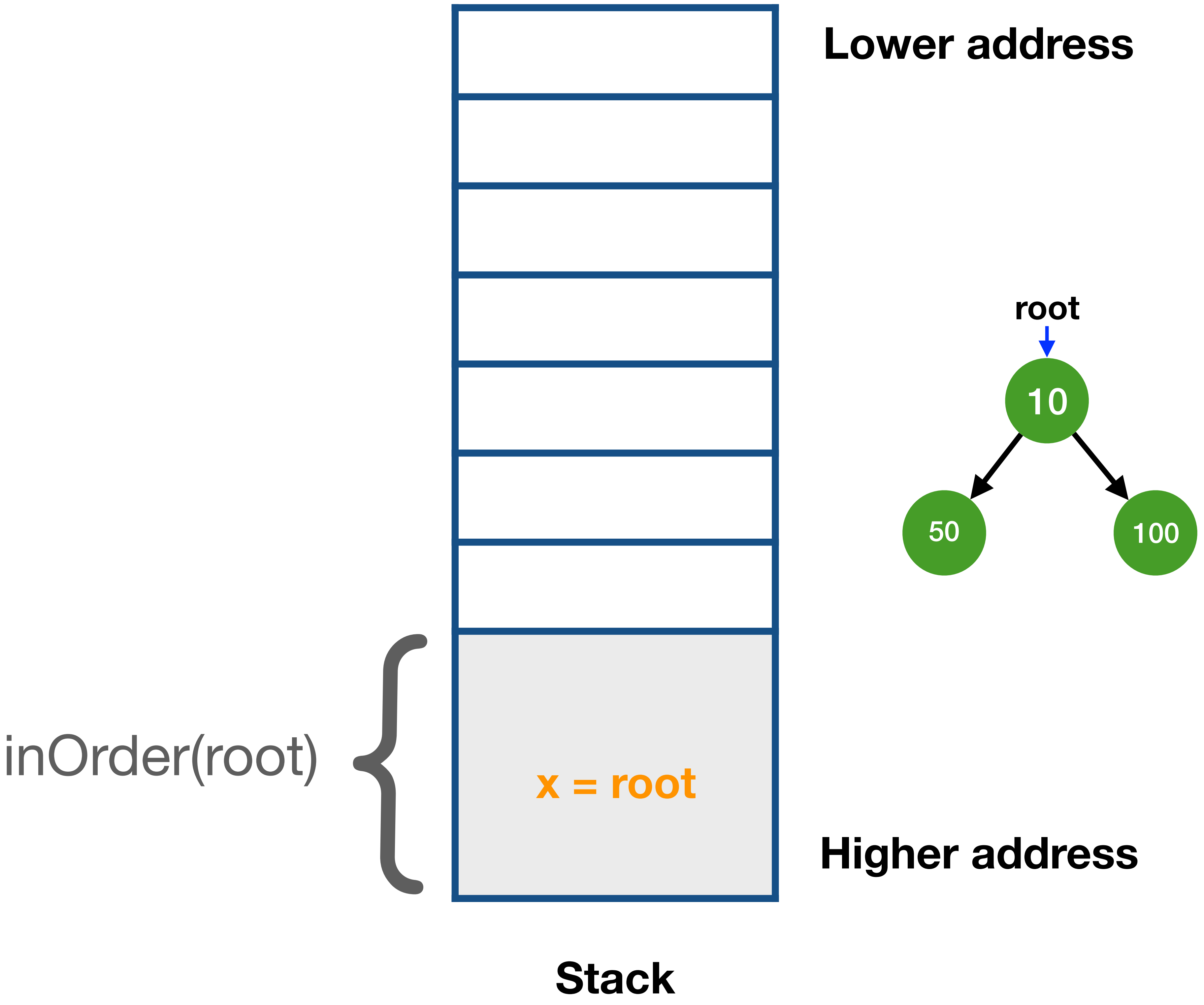
Recursion binary tree

```
inOrder(x){
 if(x != NULL){
  inOrder(x->left);
  print x->key;
  inOrder(x->right);
 }
}
```



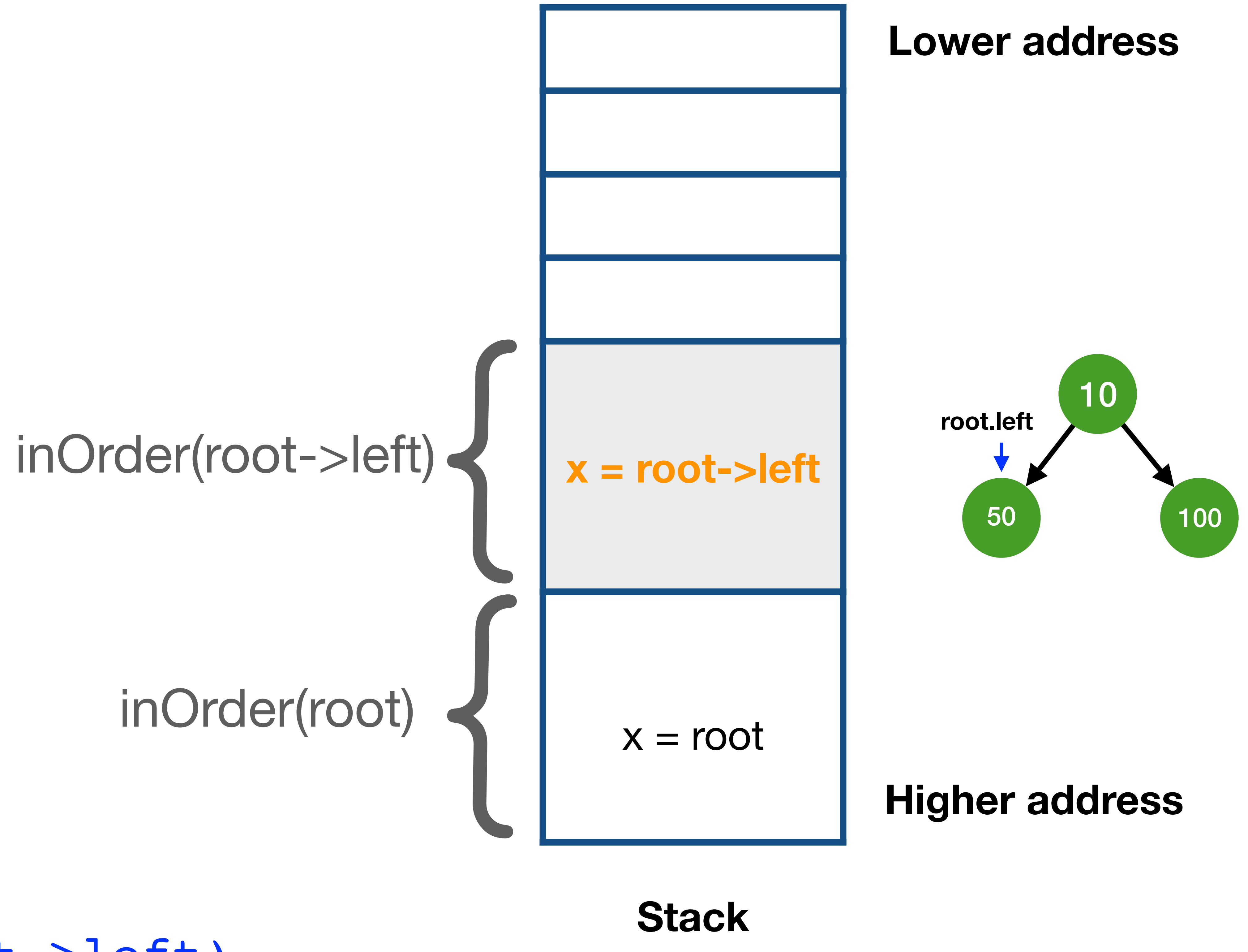
Recursion binary tree

```
x = root

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}
```



Recursion binary tree

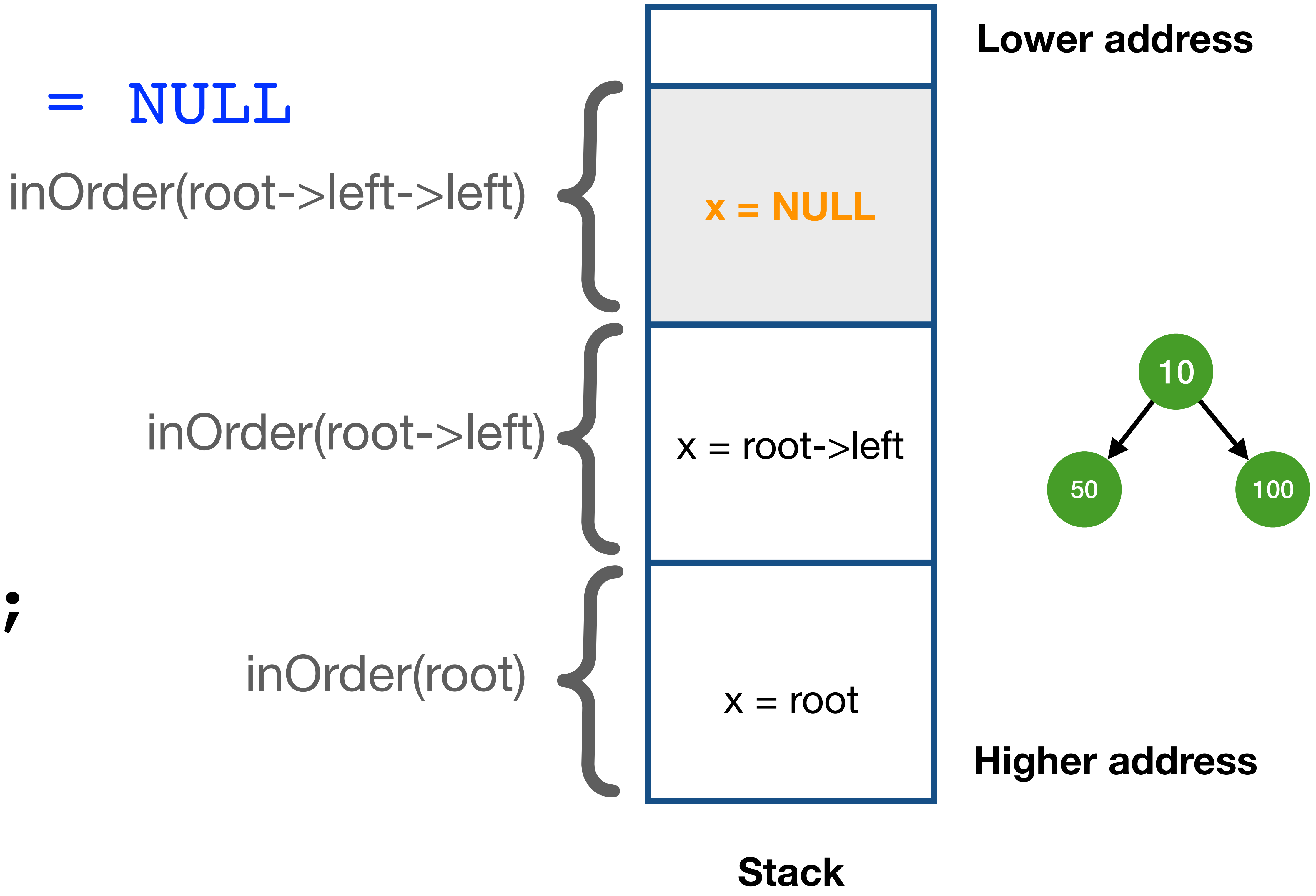
```
x = root->left

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}
```



```
inOrder(root)->inOrder(root->left)
```

Recursion binary tree

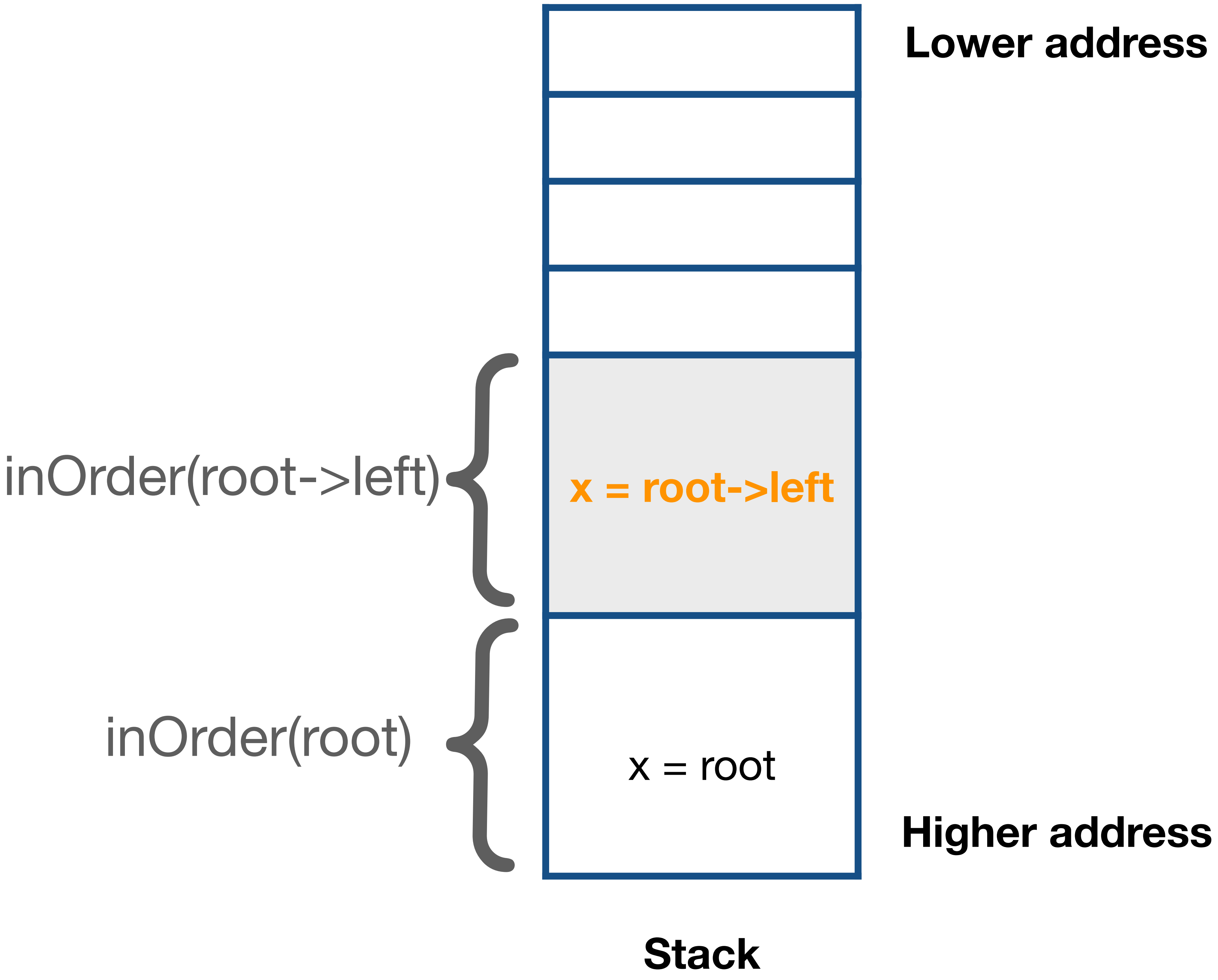
```
x = root->left->left = NULL

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}
```



`inOrder(root)->inOrder(root->left)-> inOrder(root->left->left)`

Recursion binary tree

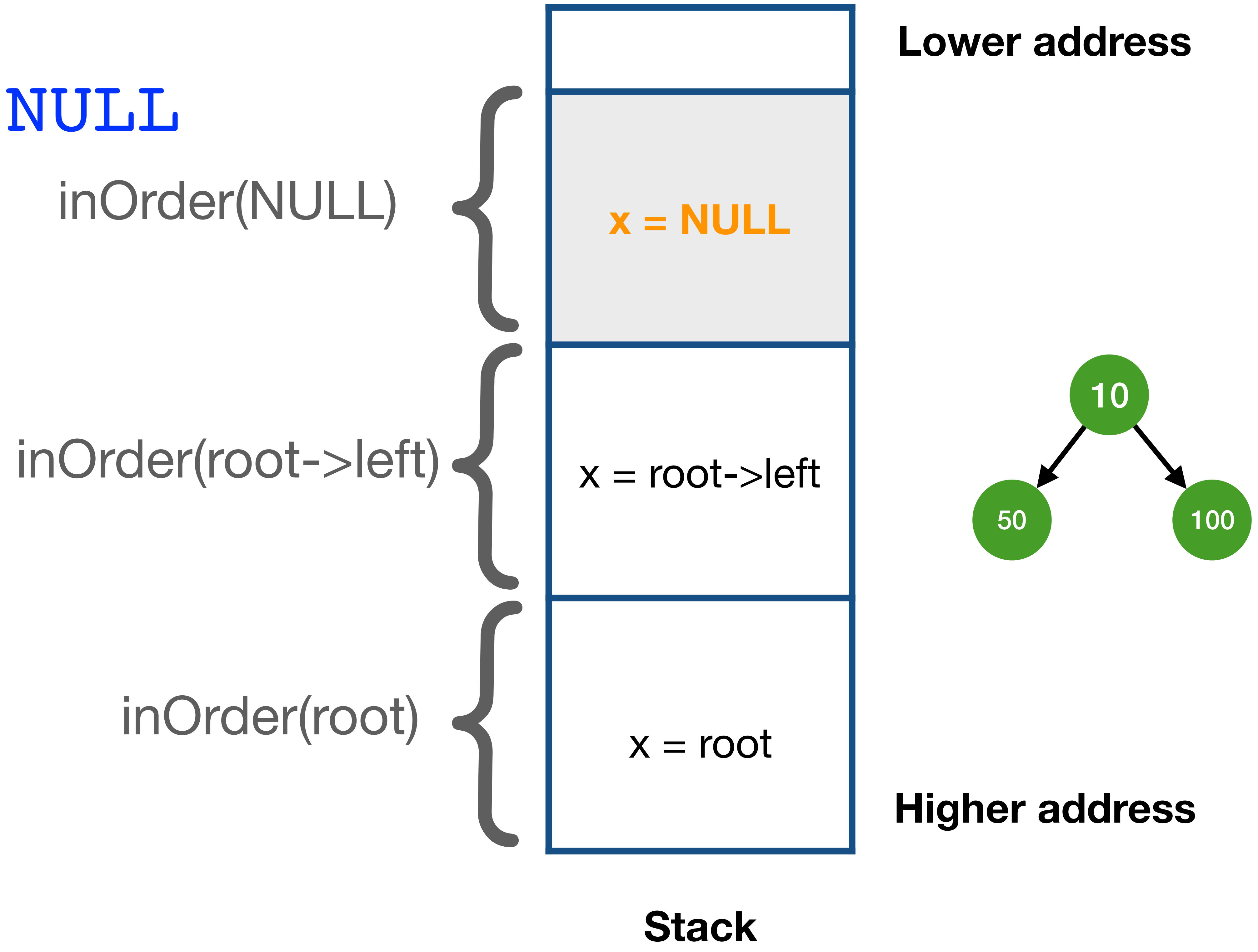
```
Display x->key : 50

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}
```



```
inOrder(root)->inOrder(root->left)
```

Recursion binary tree

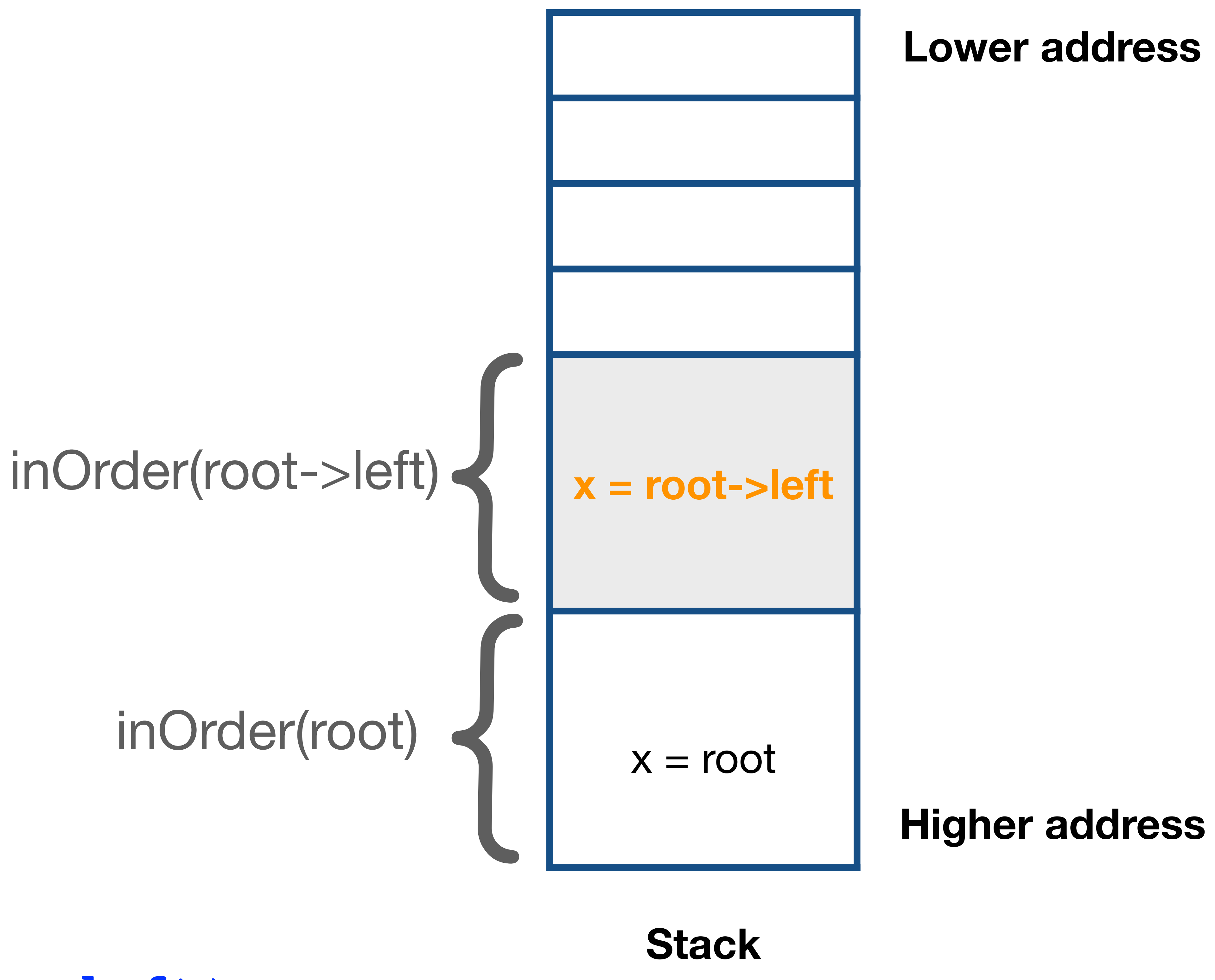
```
x = root->left->right = NULL

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}
```



`inOrder(root)->inOrder(root->left)->inOrder(root->left->right)`

Recursion binary tree

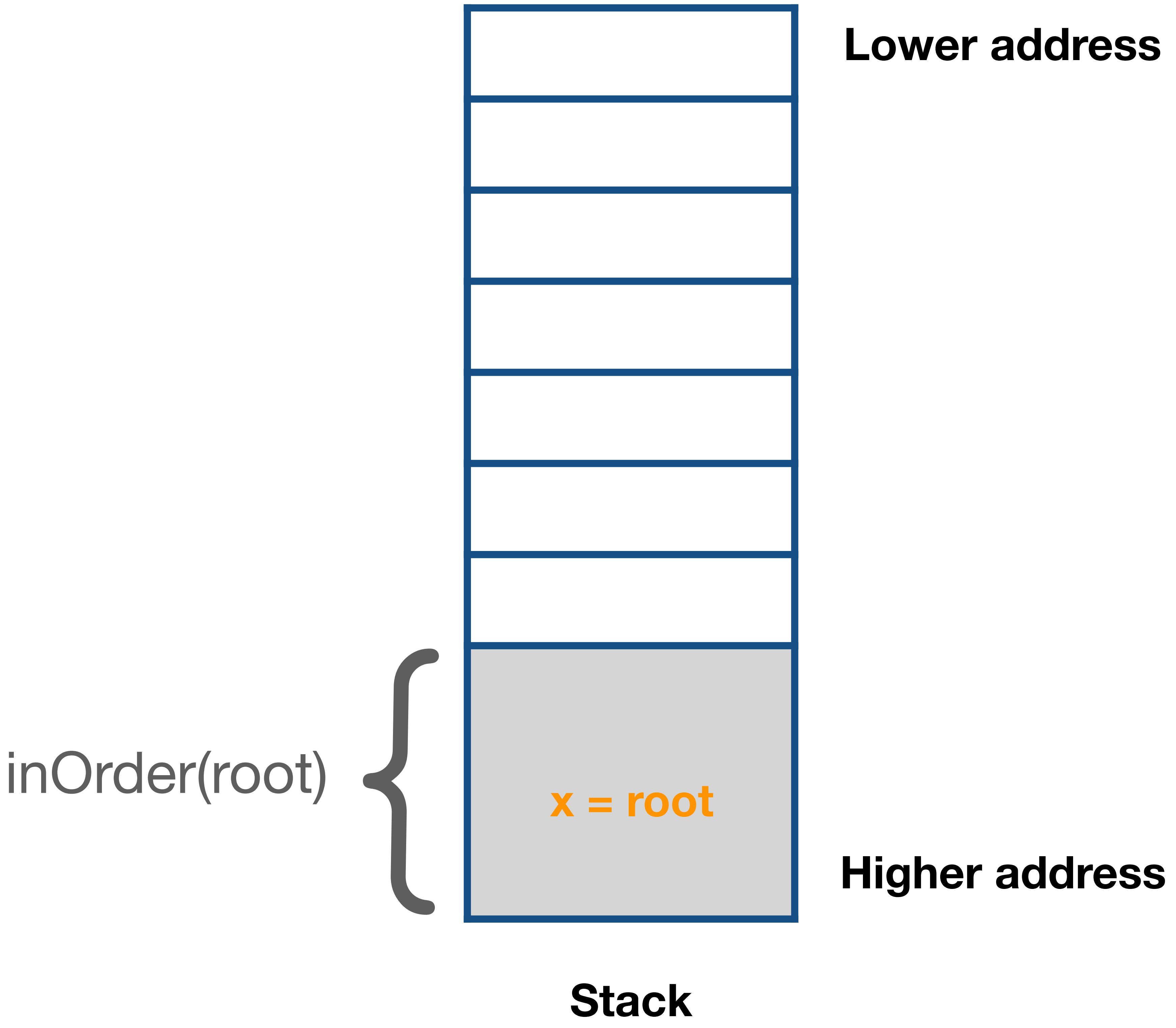
```
inOrder(x){
 if(x != NULL){
  inOrder(x->left);
  print x->key;
  inOrder(x->right);
 }
}
```



```
inOrder(root)->inOrder(root->left)
```

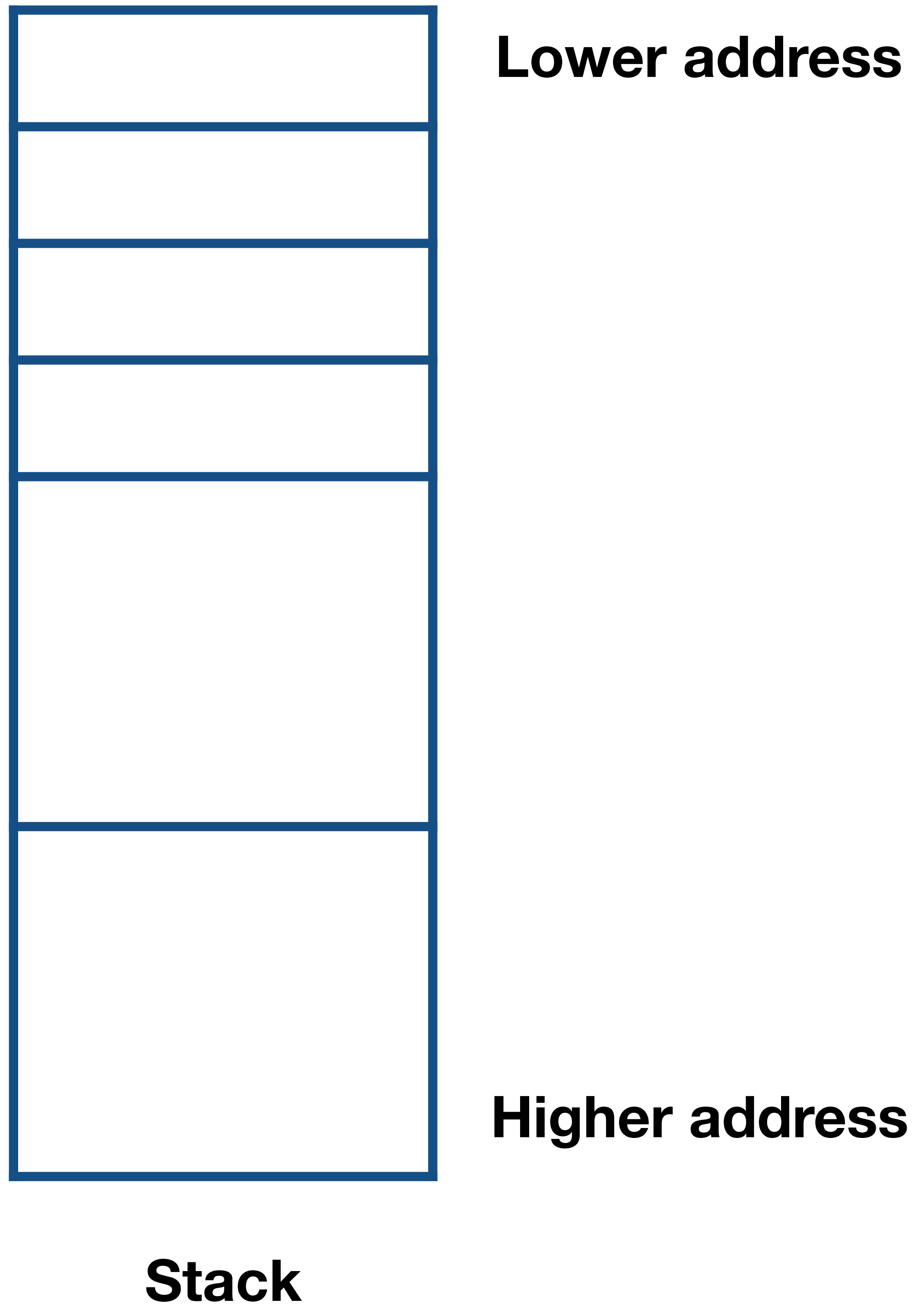
Recursion binary tree

```
Display x->key: 10;

inOrder(x){
  if(x != NULL){
    inOrder(x->left);
    print x->key;
    inOrder(x->right);
  }
}

inOrder(root)
```



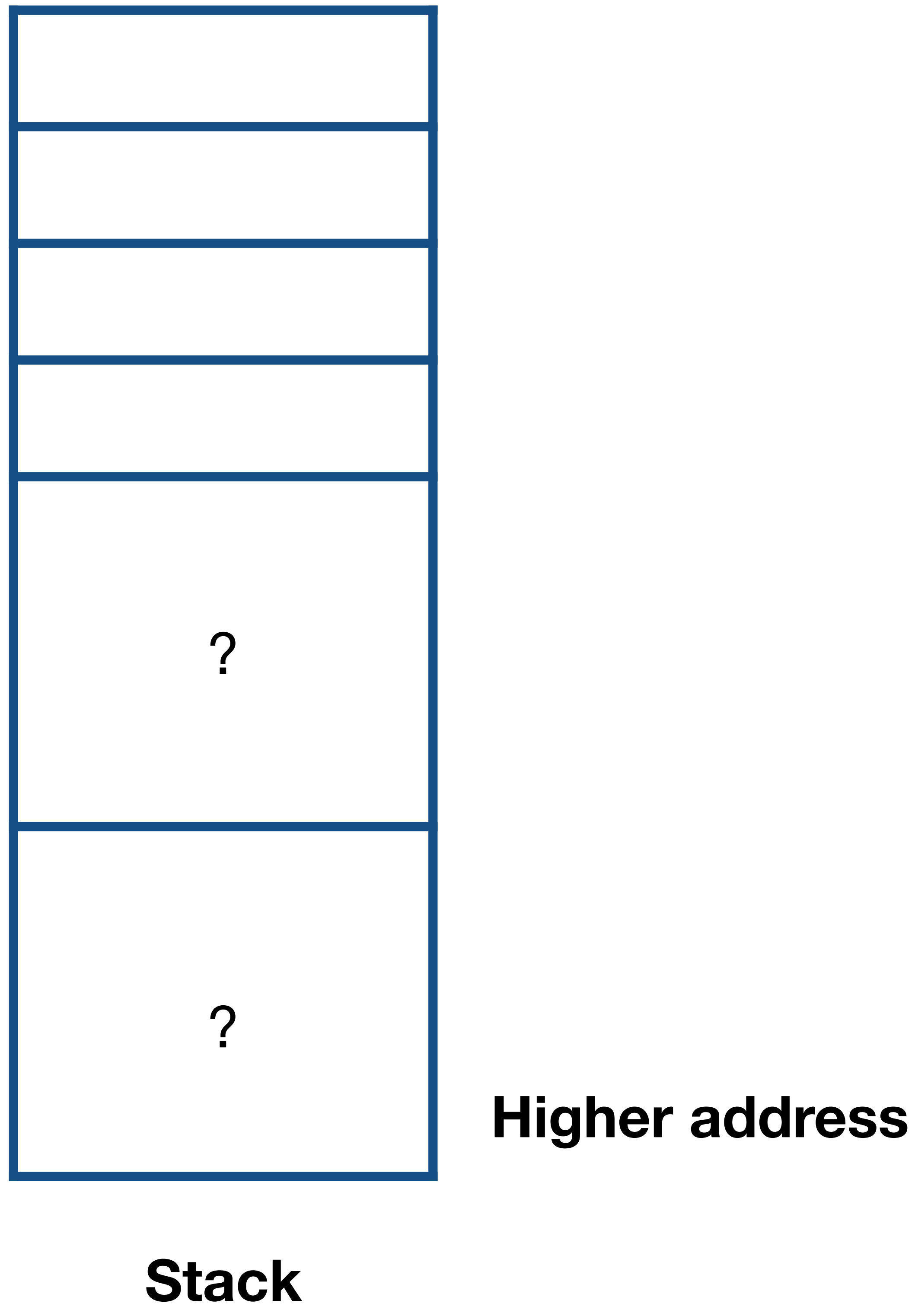
Recursion binary tree

Exercise : complete it !



Recursion DFS

Exercise : complete it !


?
?
Higher address
Stack