Recursion

Liz Willer

Finding the minimum **iteratively**

- Uses a while loop

Exercise: Write a program to find the minimum of a linked list using a while loop

min:

3

min:
3

min:

~~3~~

1

min:

~~3~~

1

min:

~~3~~

1

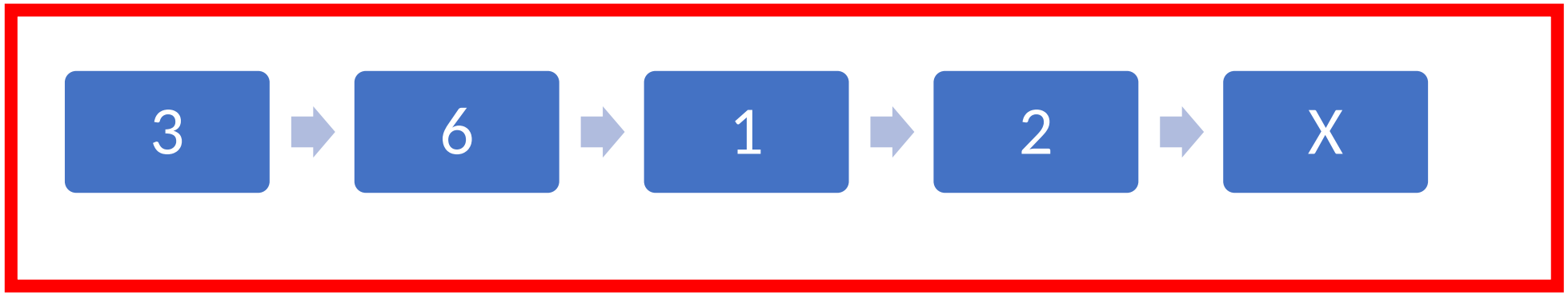
Finding the minimum **recursively**

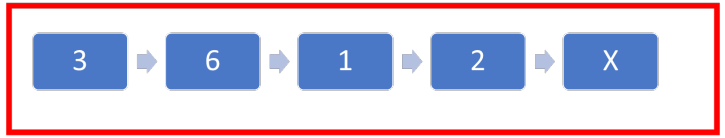
- A function calling itself

```
int rec_min(struct node *head) {
    if (head->next == NULL) {
        return head->data;
    } else {
        return minimum(rec_min(head->next), head->data);
    }
}
```

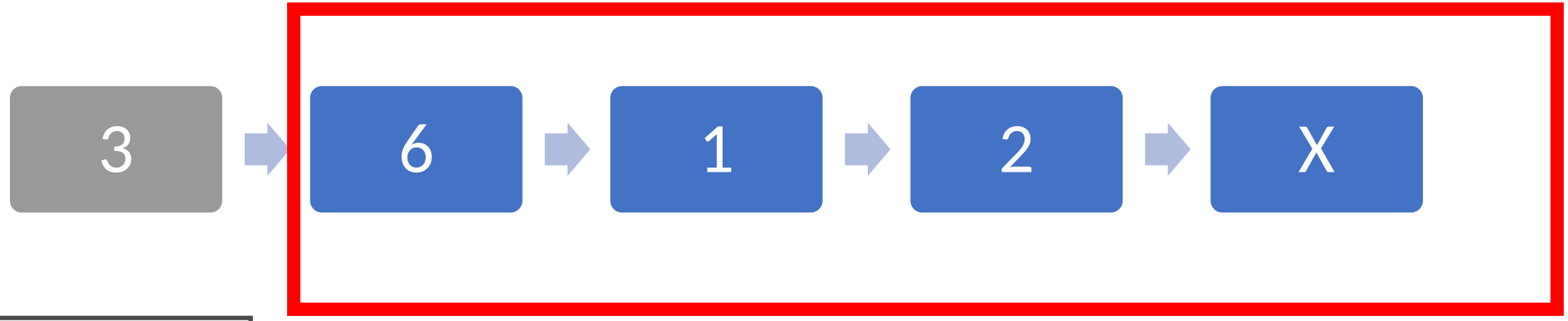
Base case

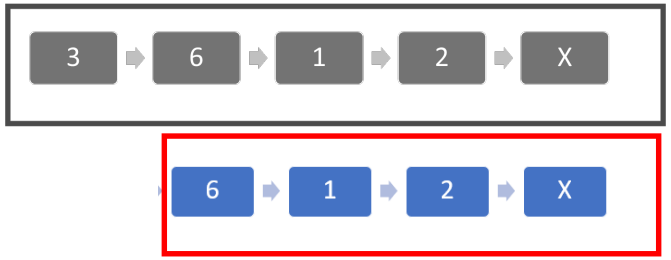
Recursive case



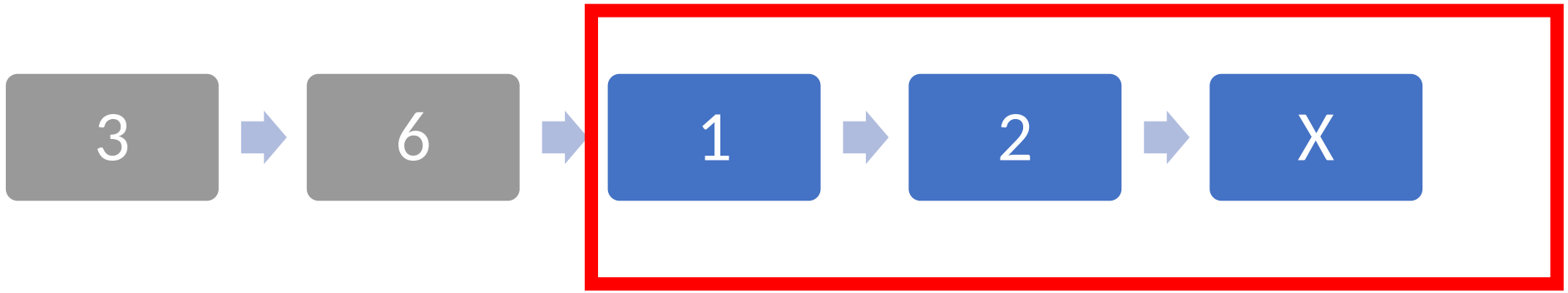


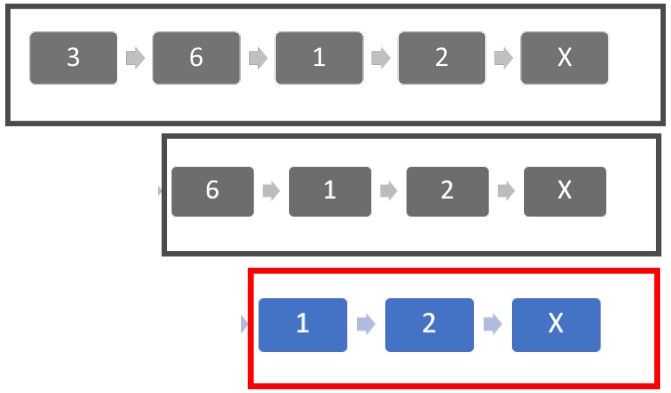

```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```



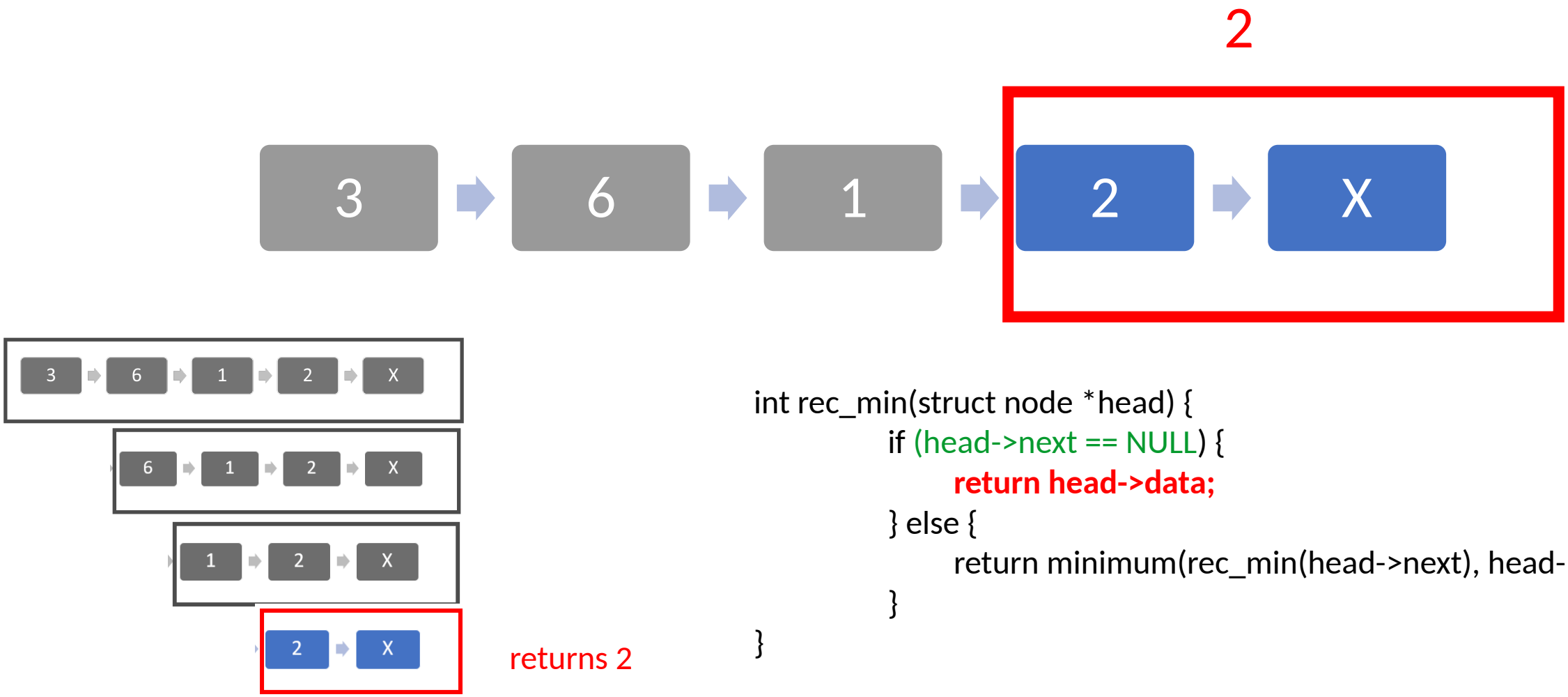



```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```

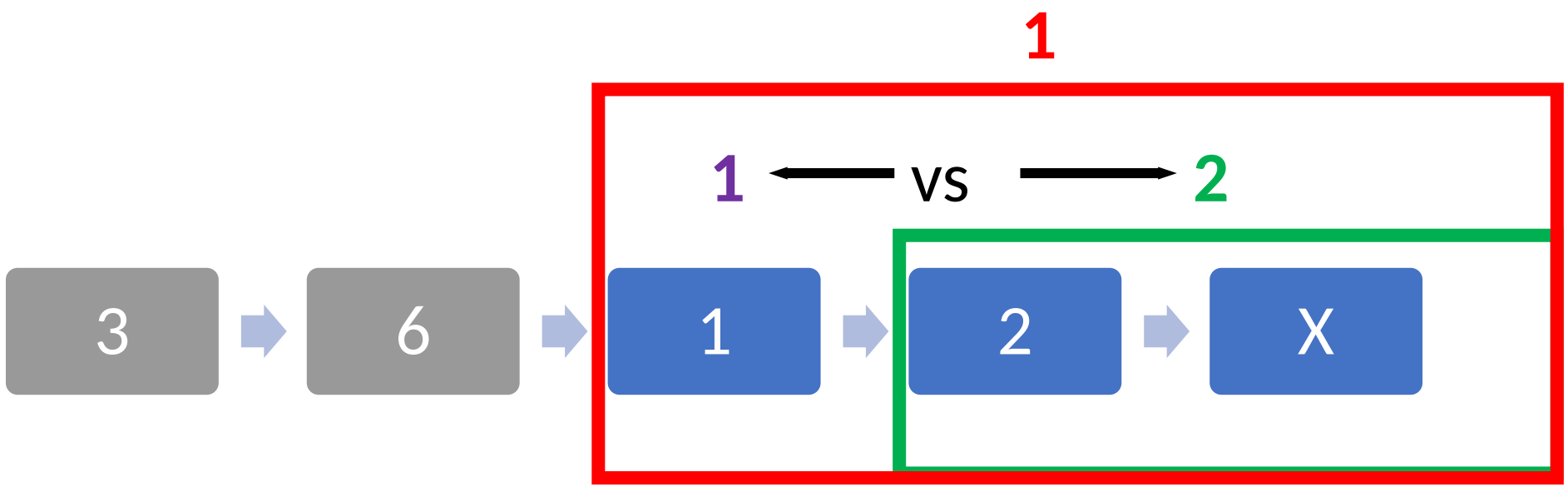


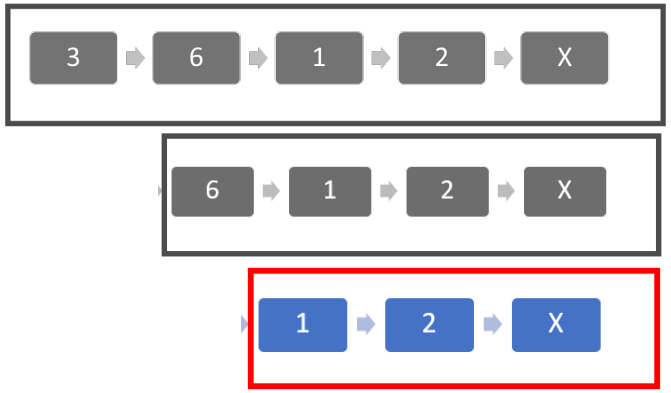



```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```



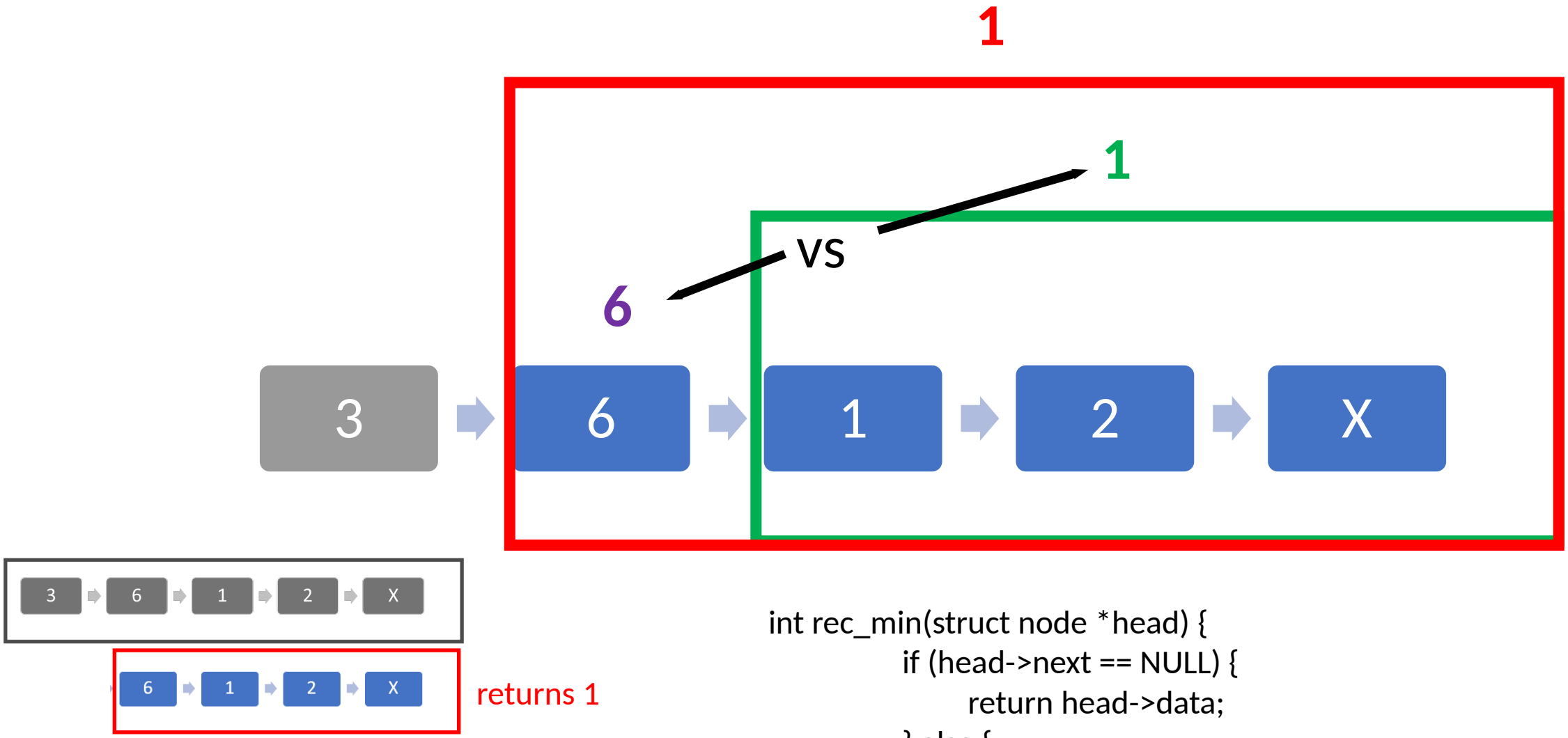

```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```



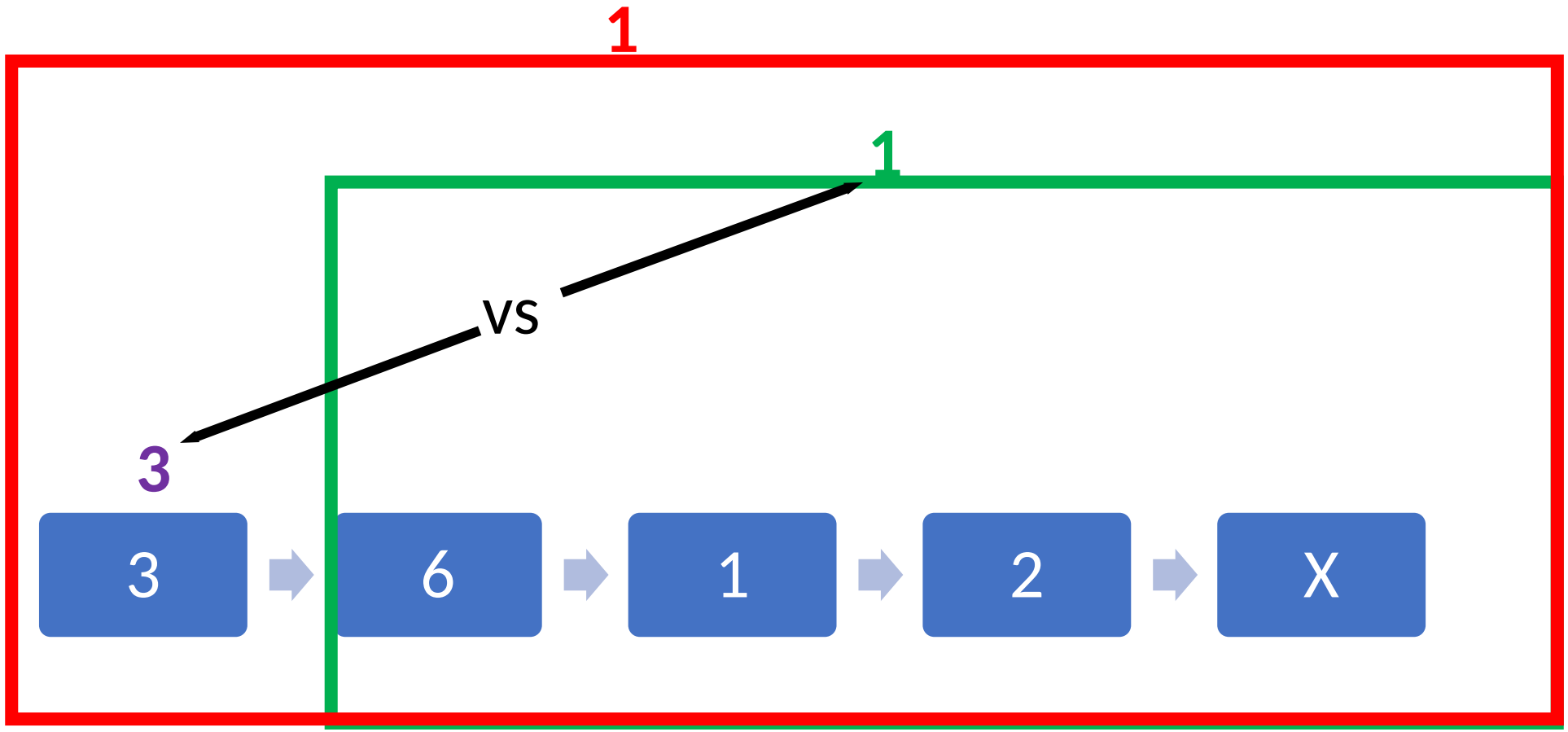


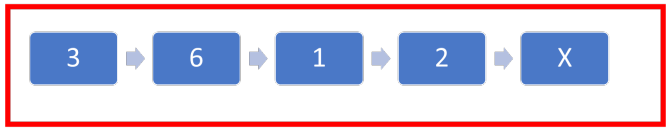
returns 1

```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```




```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```





returns 1

```
int rec_min(struct node *head) {
        if (head->next == NULL) {
                return head->data;
        } else {
                return minimum(rec_min(head->next), head->data);
        }
}
```

When do I use recursion?

1. Problems that can be broken down into smaller problems of the same sort (e.g. find the min/max of an array or list, Fibonacci).

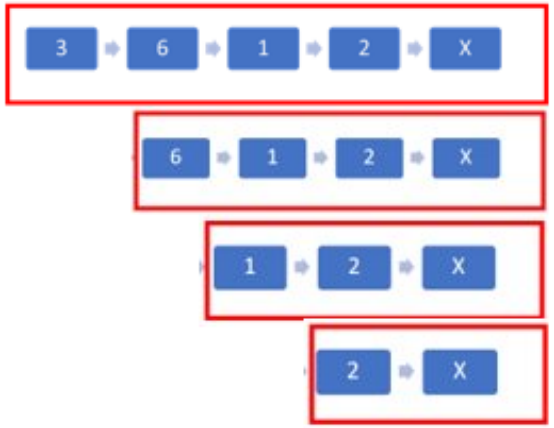
2. Problems that need a stack:

Exercise: what are some problems/algorithms that need a stack?

We can write one ourselves

OR

We can save effort by using the *call stack* instead




```c
int MAXSIZE = 8;
int stack[8];
int top = -1;

int isempty() {

   if(top == -1)
      return 1;
   else
      return 0;
}

int isfull() {

   if(top == MAXSIZE)
      return 1;
   else
      return 0;
}

int peek() {
   return stack[top];
}

int pop() {
   int data;

   if(!isempty()) {
      data = stack[top];
      top = top - 1;
      return data;
   } else {
      printf("Could not retrieve data, Stack is empty.\n");
   }
}

int push(int data) {

   if(!isfull()) {
      top = top + 1;
      stack[top] = data;
   } else {
      printf("Could not insert data, Stack is full.\n");
   }
}
```