```
procedure concat(a: Node, b: Node) returns (res: Node)
  requires lseg(a, null) * lseg(b, null);
  ensures lseg(res, null);
{
  if (a == null)
    return b;

  Node curr := a;

  while (curr.next != null)
    invariant curr != null * lseg(a, curr) * lseg(curr, null);
    curr := curr.next;

  curr.next := b;
  return a;
}
```

pre / postconditions

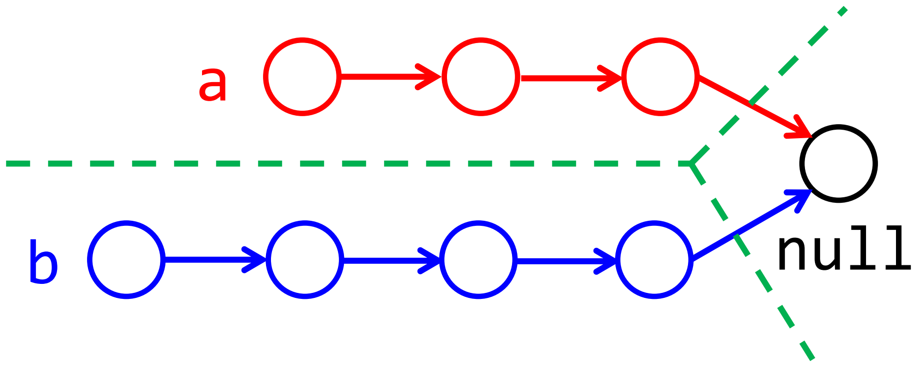
loop invariants

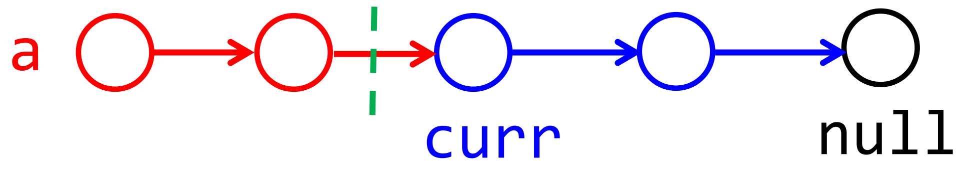
```
procedure concat(a: Node, b: Node) returns (res: Node)
  requires lseg(a, null) * lseg(b, null);
  ensures lseg(res, null);
{
  if (a == null)
    return b;

  Node curr := a;

  while (curr.next != null)
    invariant curr != null * lseg(a, curr) * lseg(curr, null);
    curr := curr.next;

  curr.next := b;
  return a;
}
```



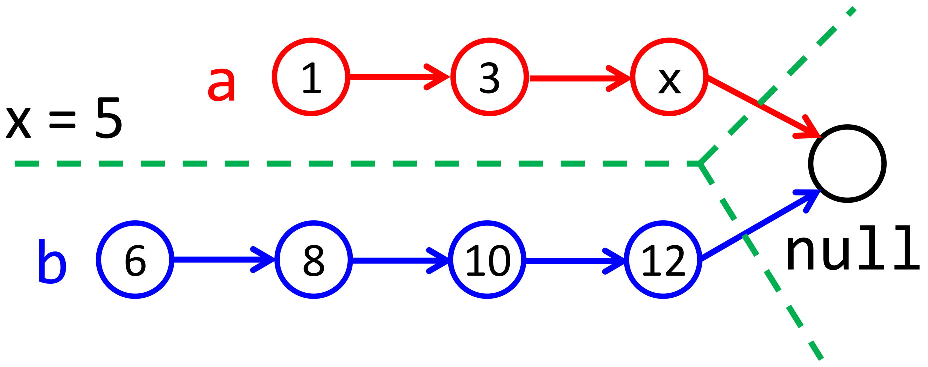


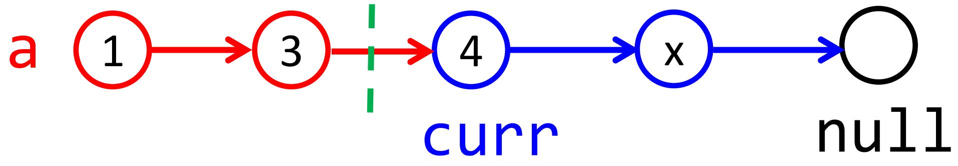
```
procedure concat(a: Node, b: Node) returns (res: Node)
  requires lsleg(a, null, x) * uslseg(b, null, x);
  ensures slseg(res, null);
{
  if (a == null)
    return b;

  Node curr := a;

  while (curr.next != null)
    invariant curr != null;
    invariant lslseg(a, curr, curr.data) * lslseg(curr, null, x);
    curr := curr.next;

  curr.next := b;
  return a;
}
```





```
procedure concat(a: Node, b: Node) returns (res: Node)
  requires lseg(a, null) * lseg(b, null);
  ensures lseg(res, null);
{
  if (a == null)
    return b;

  Node curr := a;

  while (curr.next != null)
    invariant curr != null * lseg(a, curr) * lseg(curr, null);
    curr := curr.next;

  curr.next := b;
  return a;
}
```