

# SQLServerFast.com

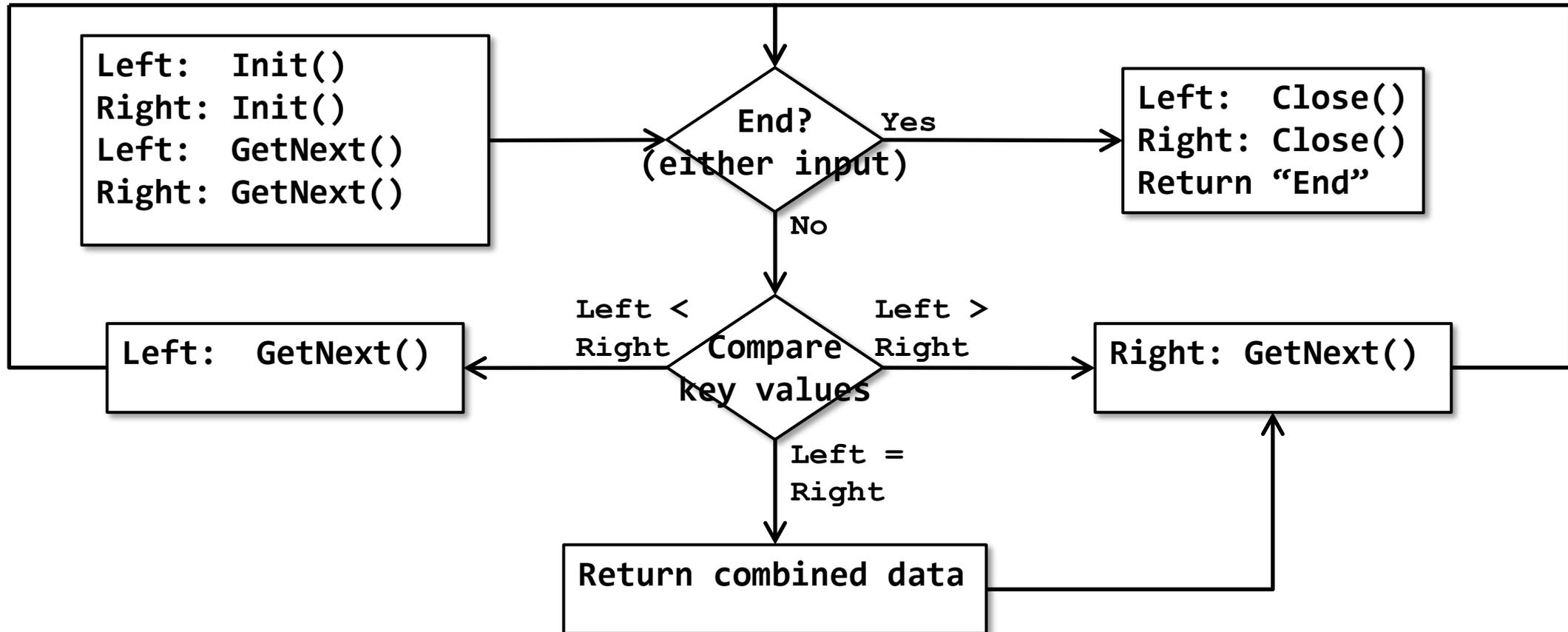
## Execution Plan Video Training

Block 3: Combining data

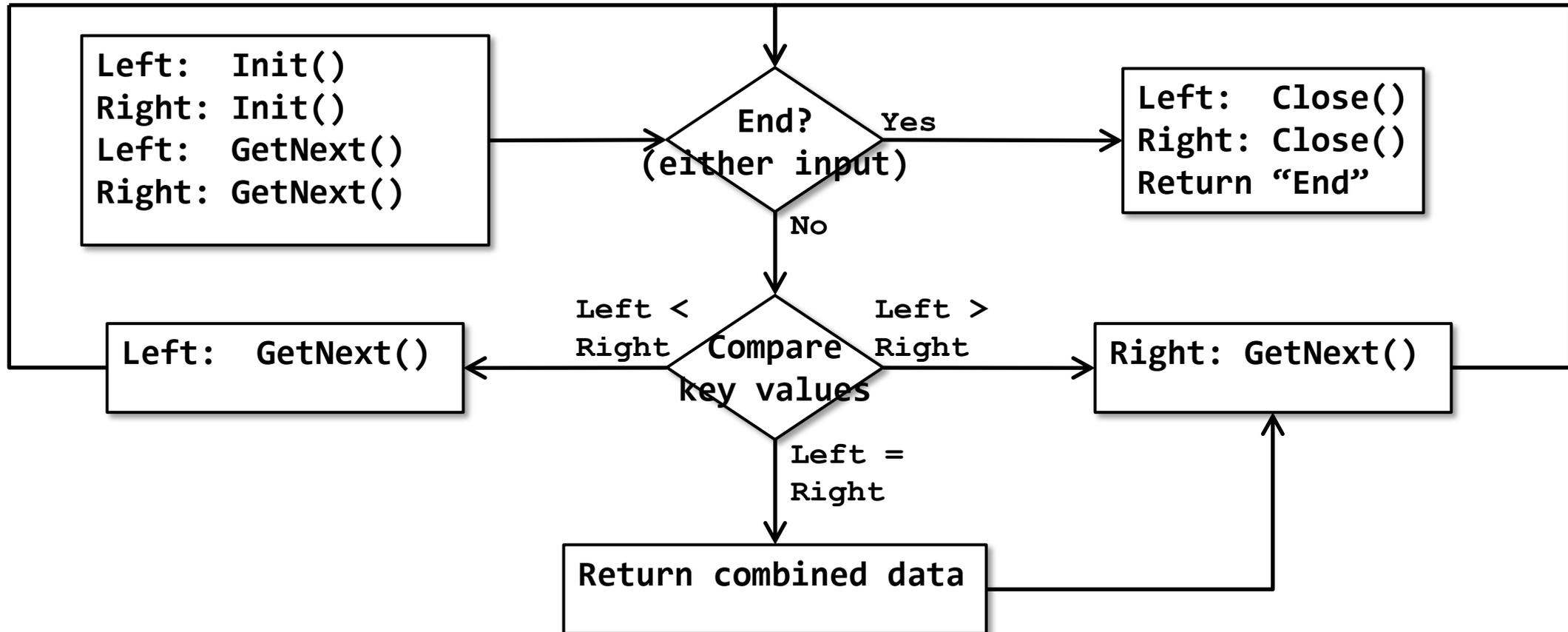
Level: Advanced

Chapter 2: Merge Join (advanced)

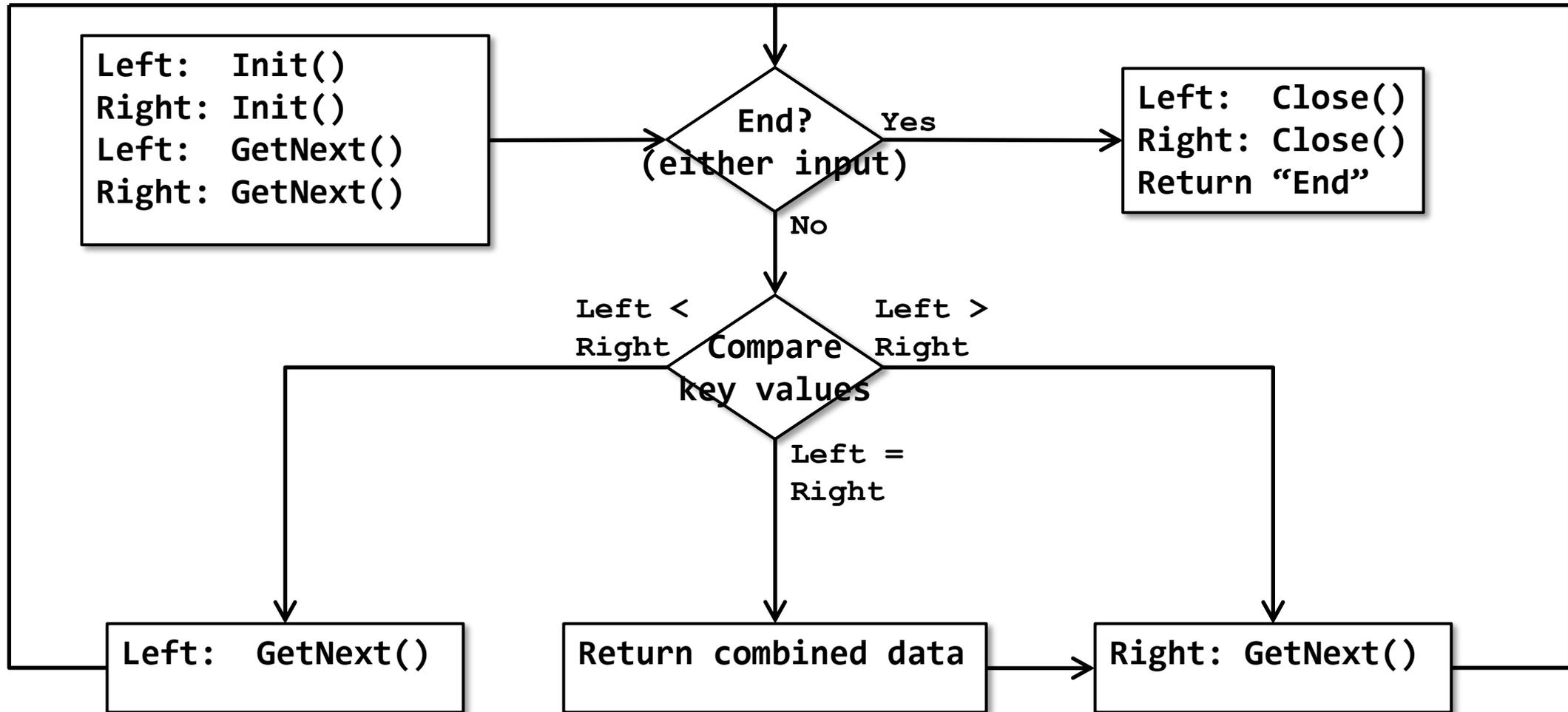
# Merge Join (inner join, one-to-many)



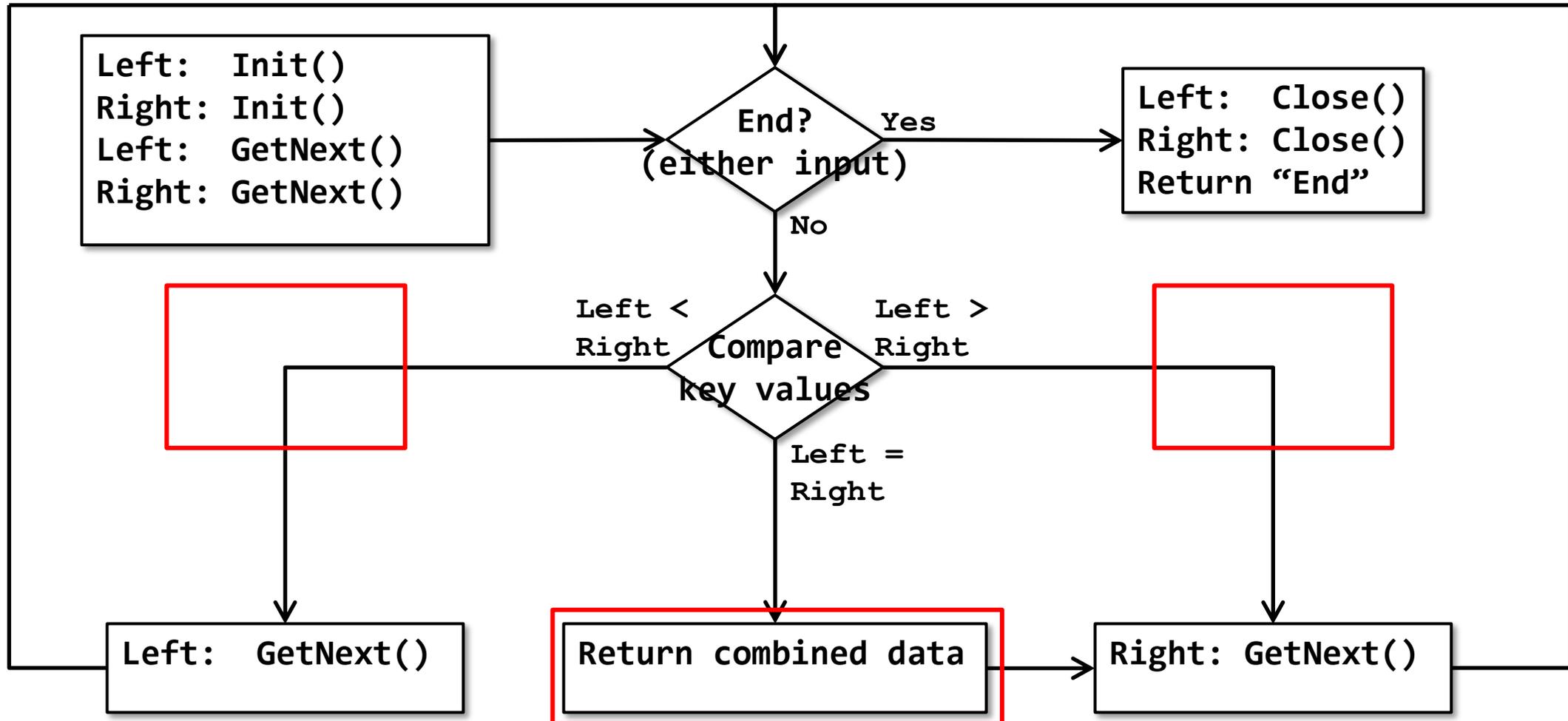
# Merge Join (inner join, one-to-many)



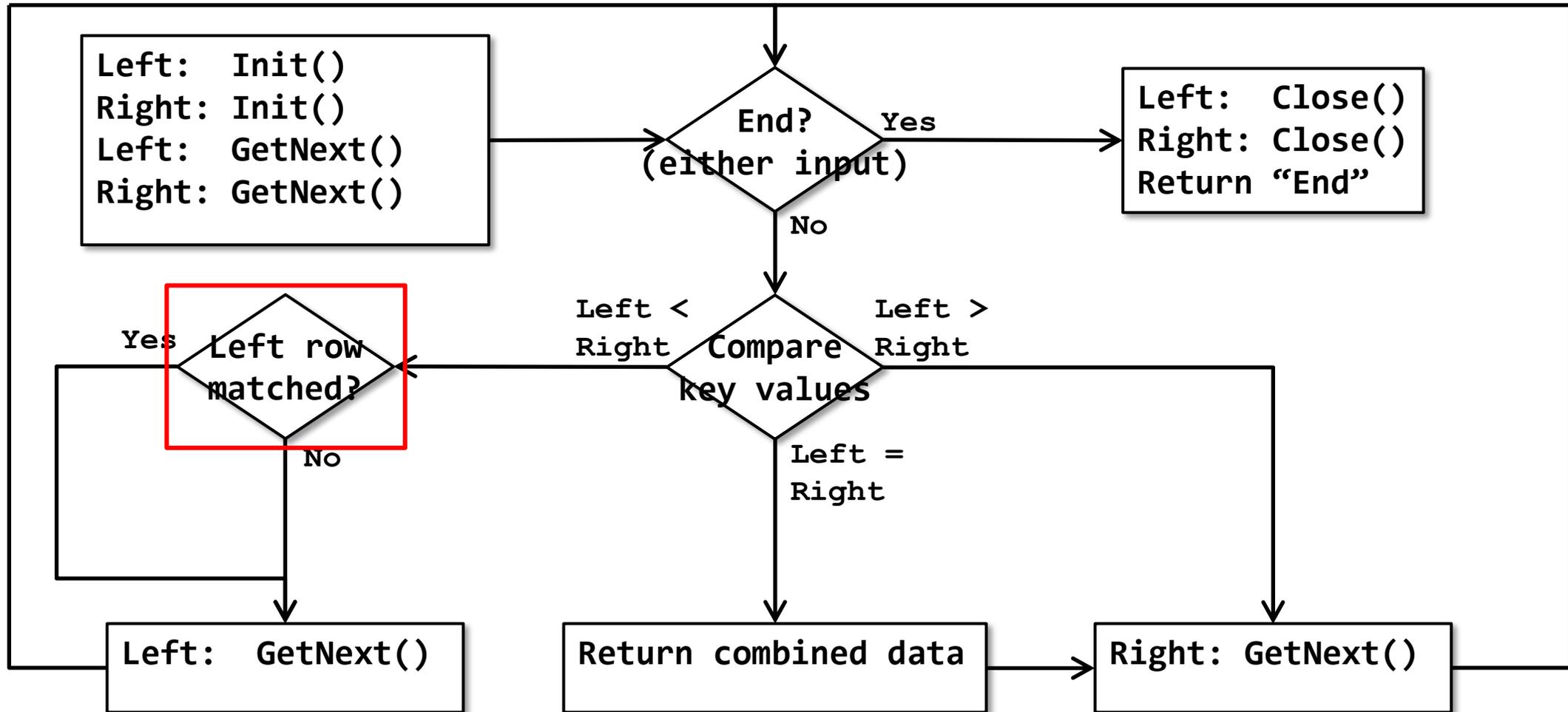
# Merge Join (inner join, one-to-many)



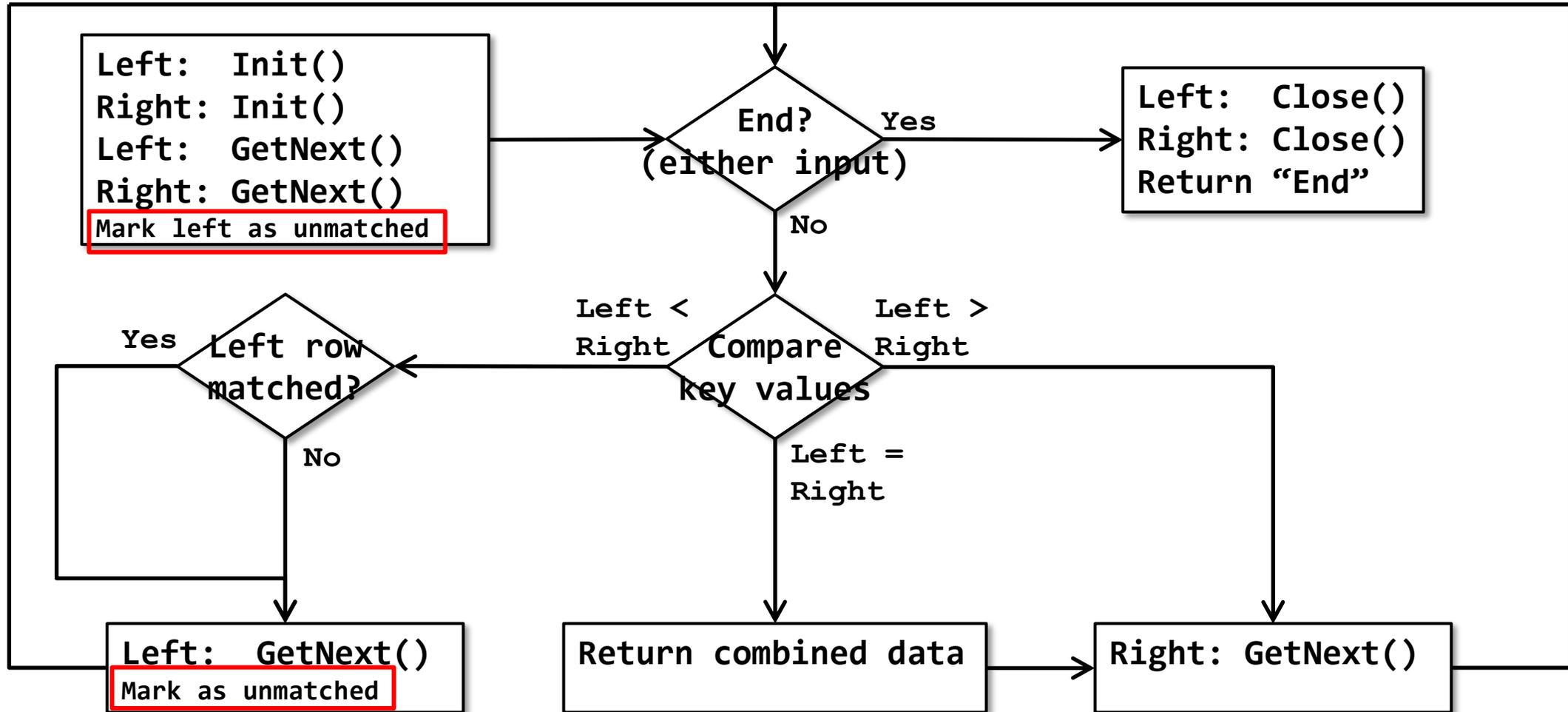
# Merge Join (inner to left outer)



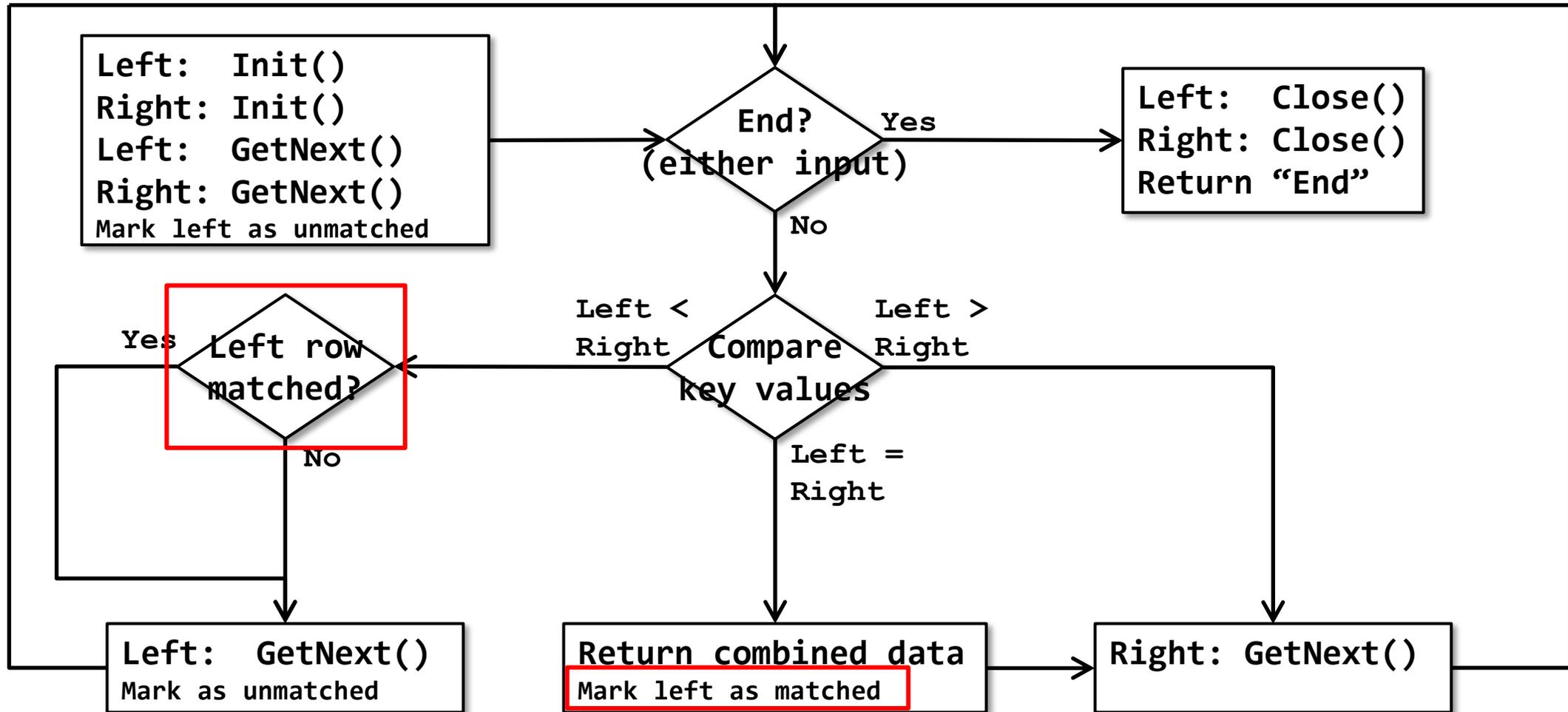
# Merge Join (inner to left outer)



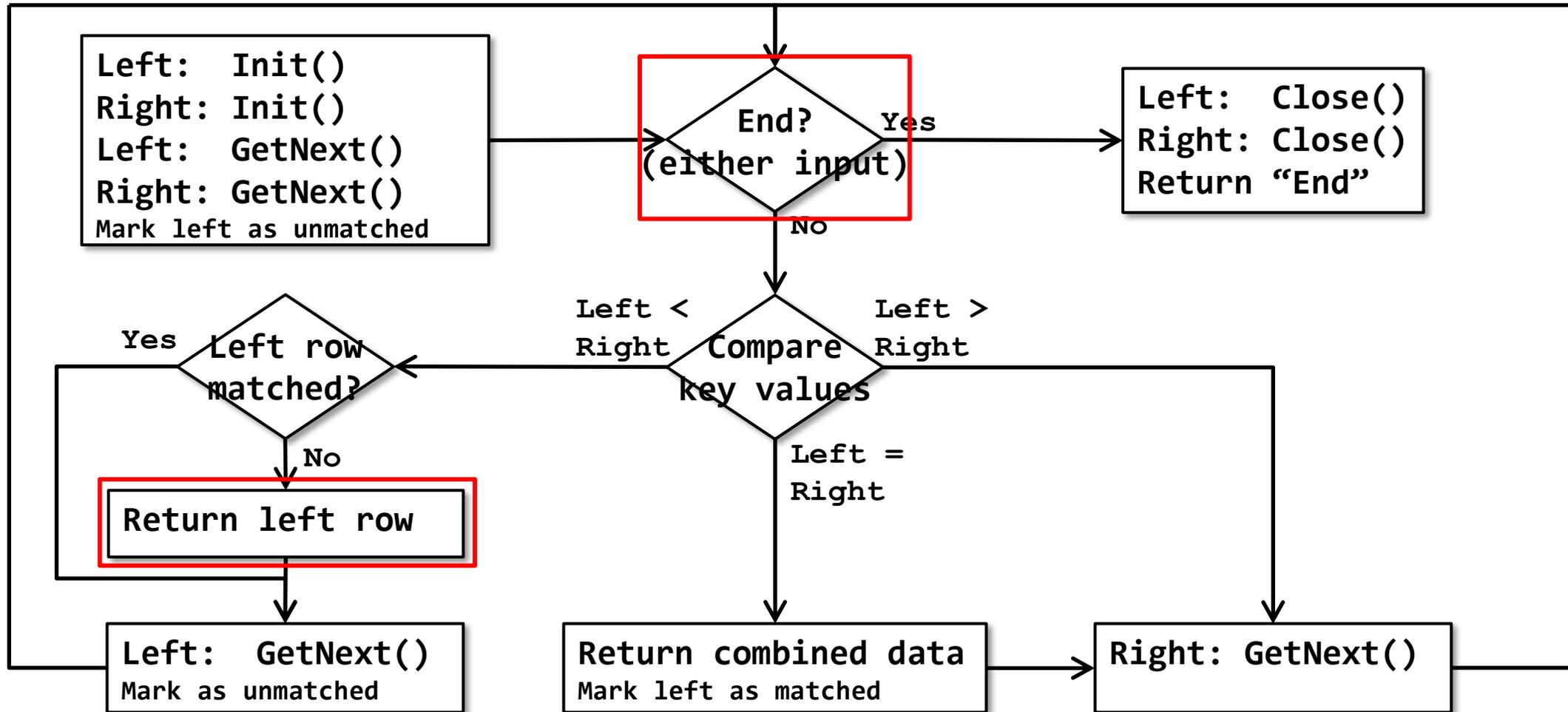
# Merge Join (inner to left outer)



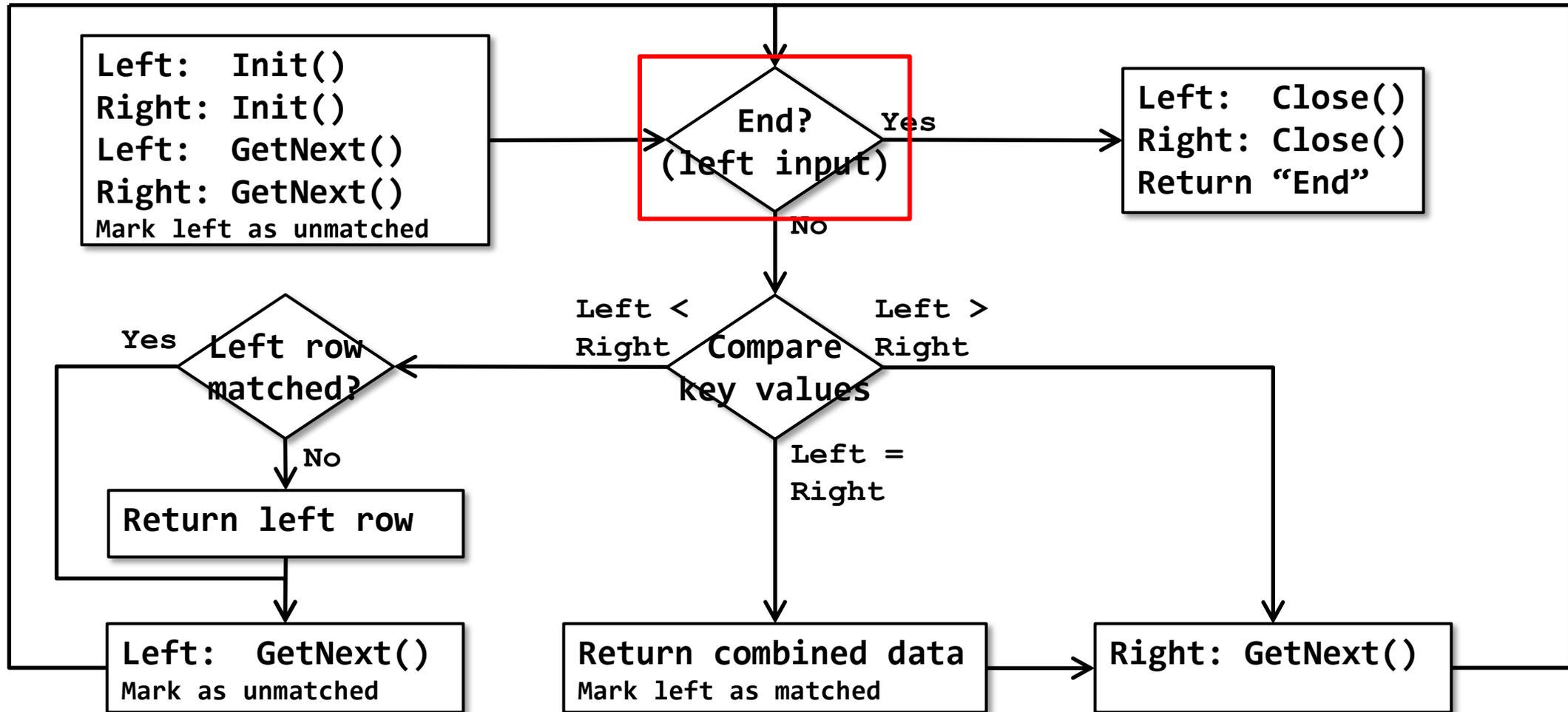
# Merge Join (inner to left outer)



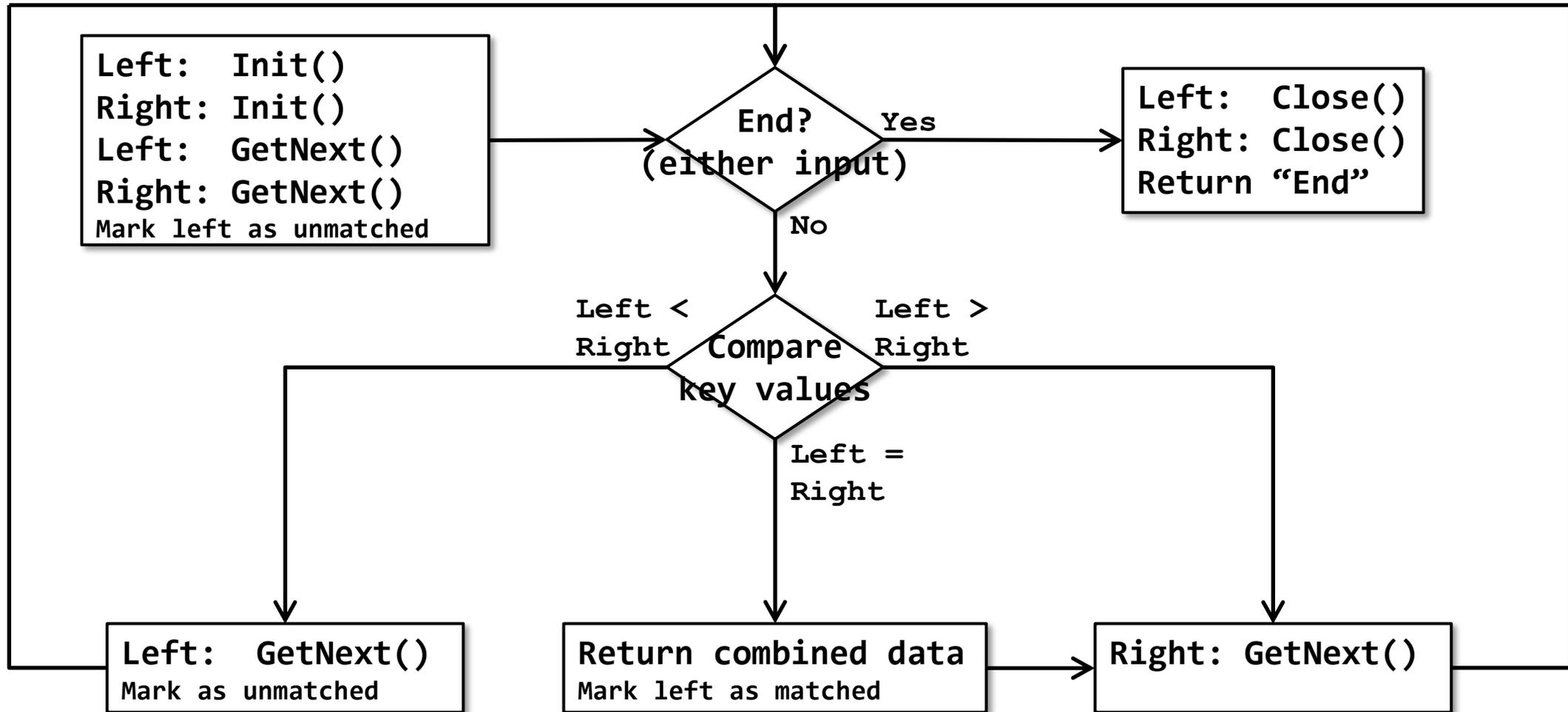
# Merge Join (inner to left outer)



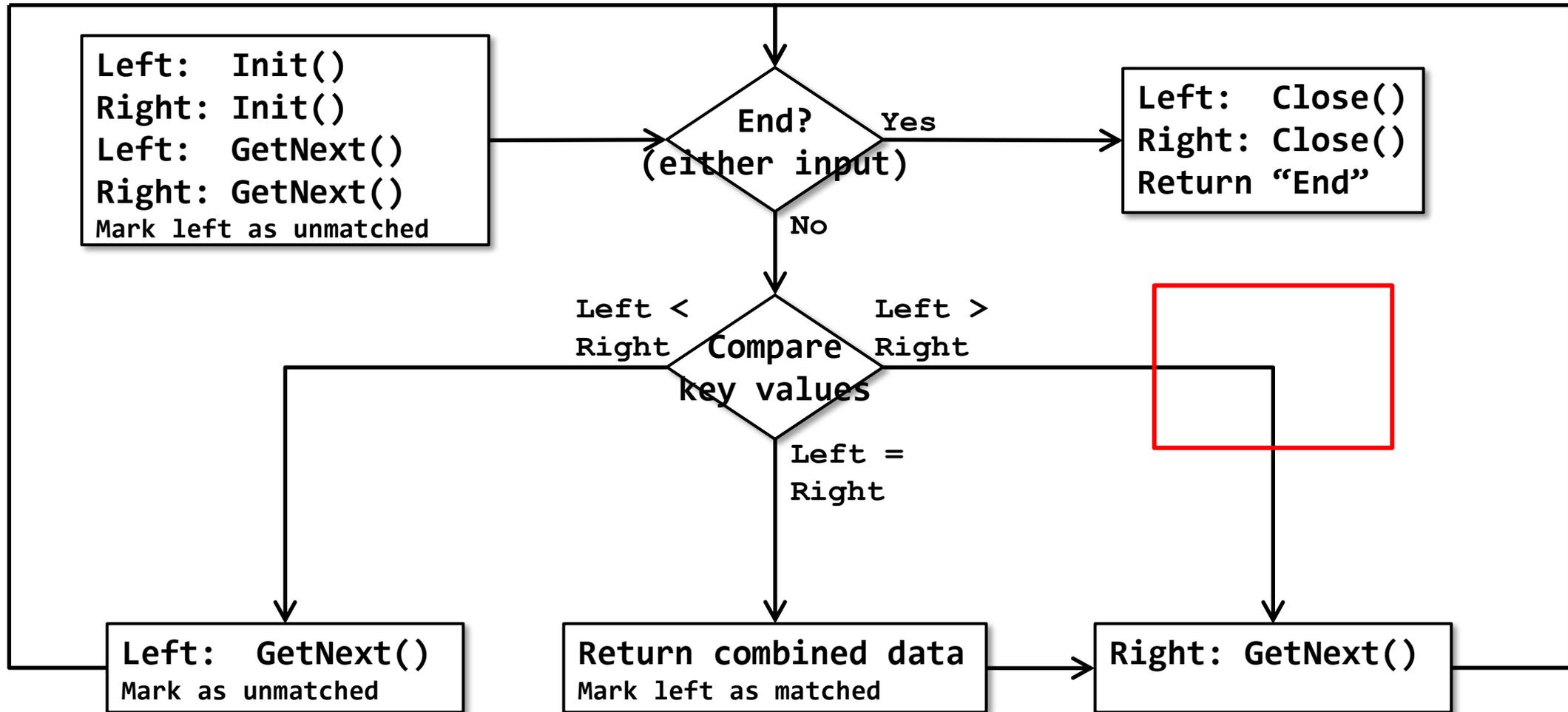
# Merge Join (left outer join, one to many)



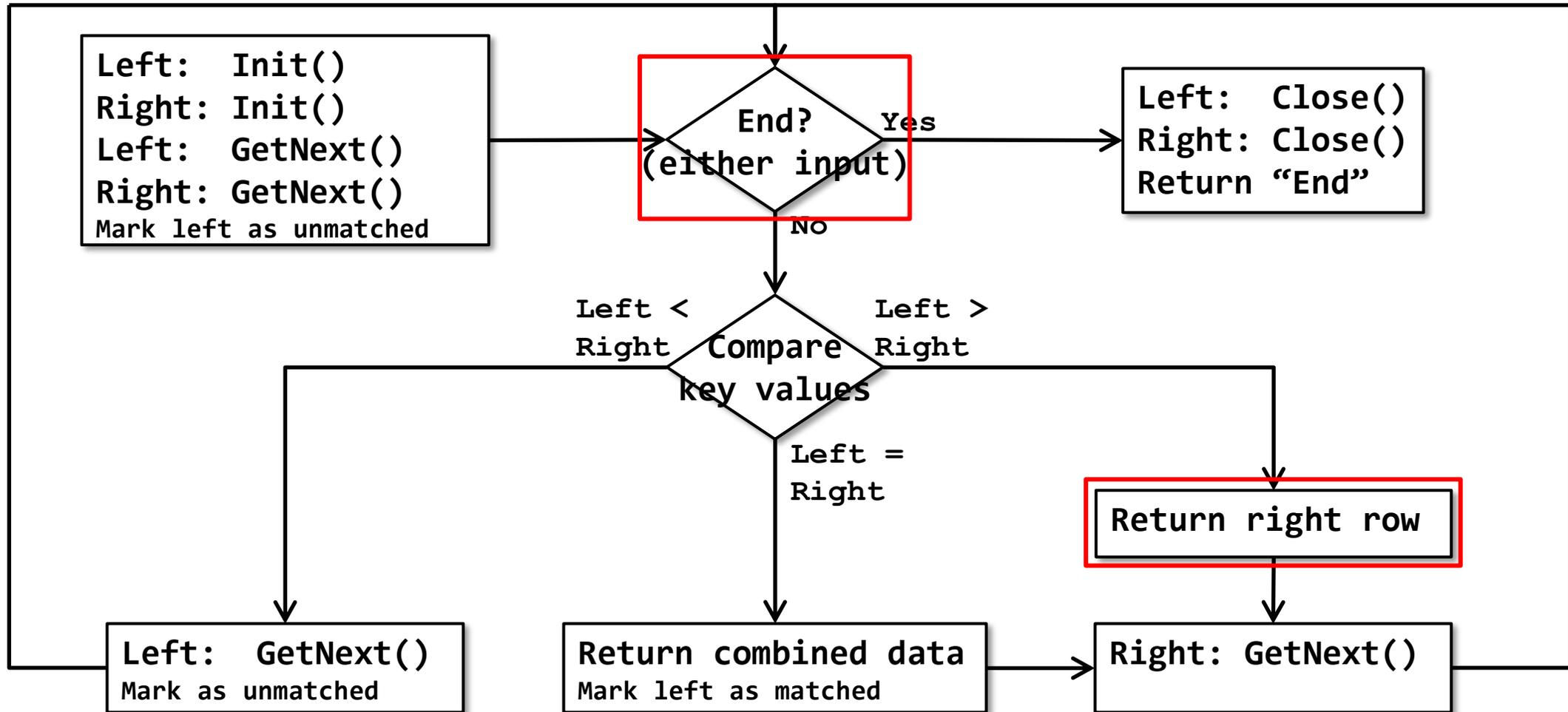
# Merge Join (inner join, one-to-many)



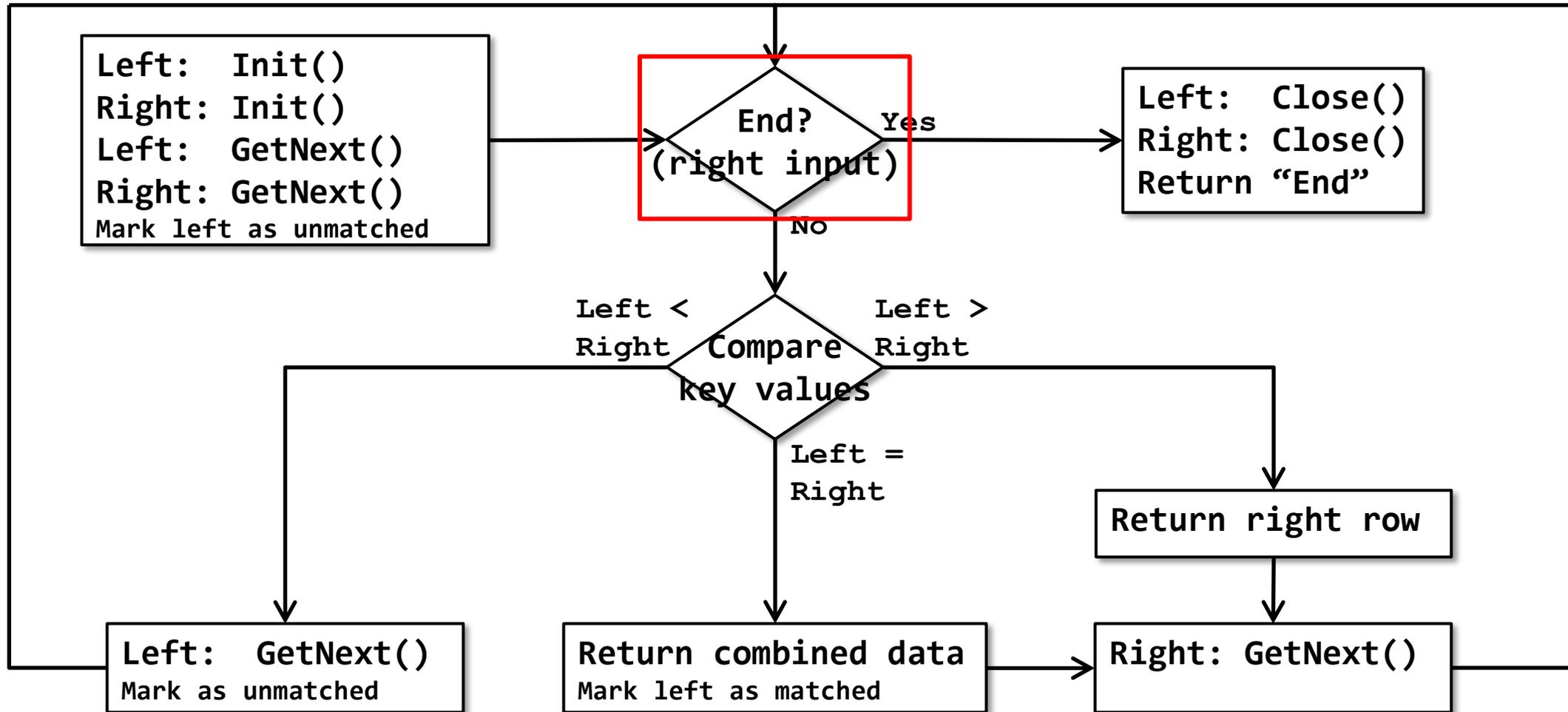
# Merge Join (inner to right outer)



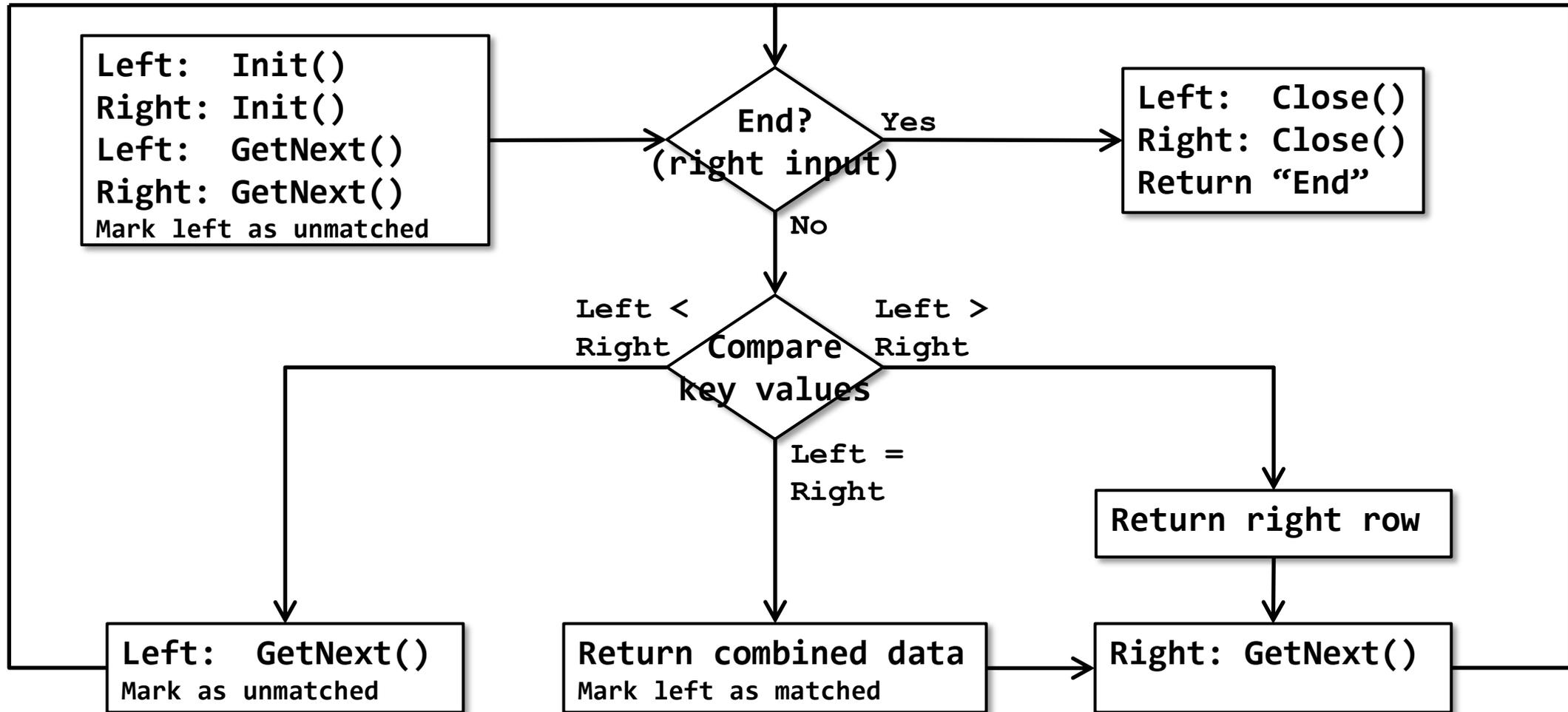
# Merge Join (inner to right outer)



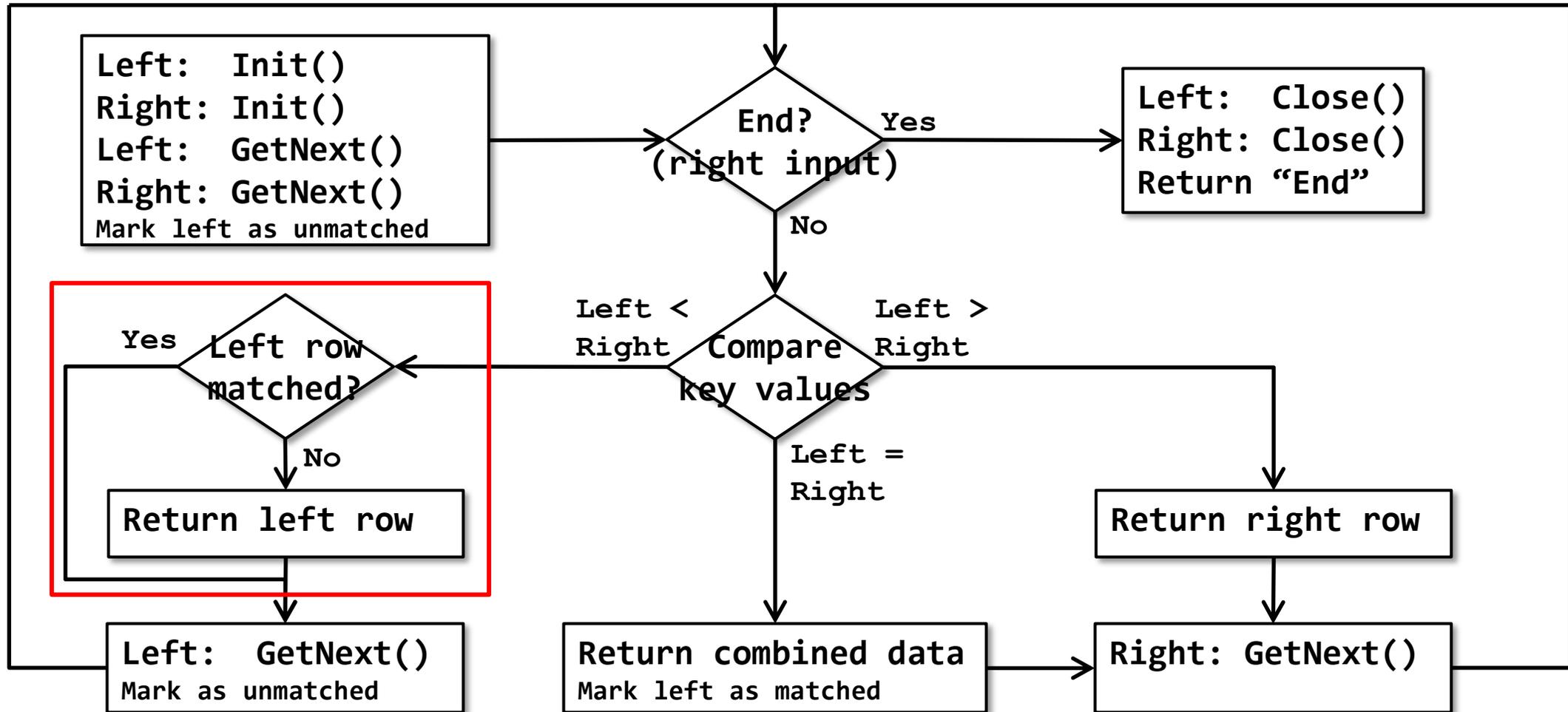
# Merge Join (right outer join, one to many)



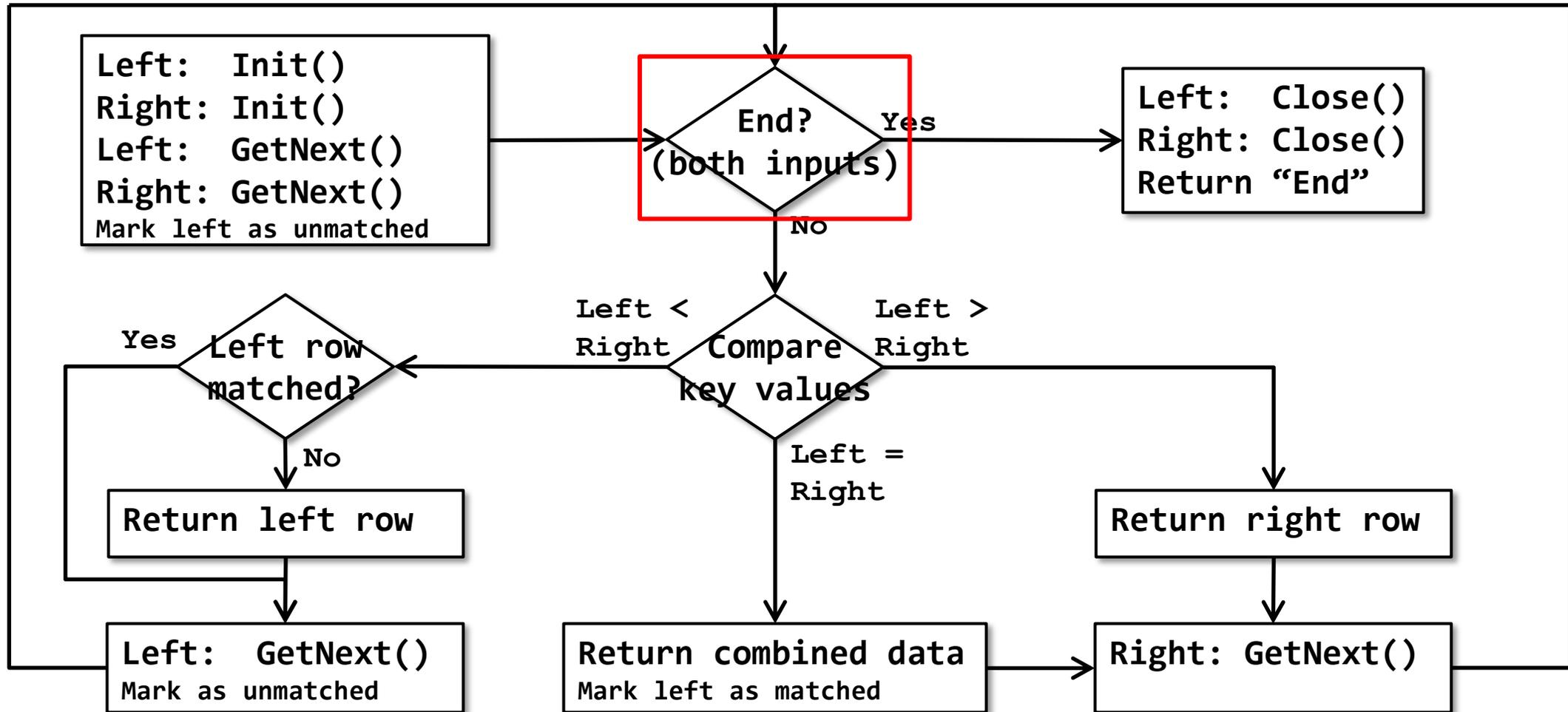
# Merge Join (right outer to full outer)



# Merge Join (right outer to full outer)



# Merge Join (full outer join, one to many)



# Merge Join

## Merge Join

Retains order of both inputs



# Merge Join

## Merge Join

Retains order of both inputs *for Inner Join*



# Merge Join

## Merge Join

Retains order of both inputs for Inner Join

Retains order of left input for Left Outer Join

Left-retained rows are out of order for columns from right input

Retains order of left input for Right Outer Join

Right-retained rows are out of order for columns from left input

Retains order of none of the inputs for Full Outer Join

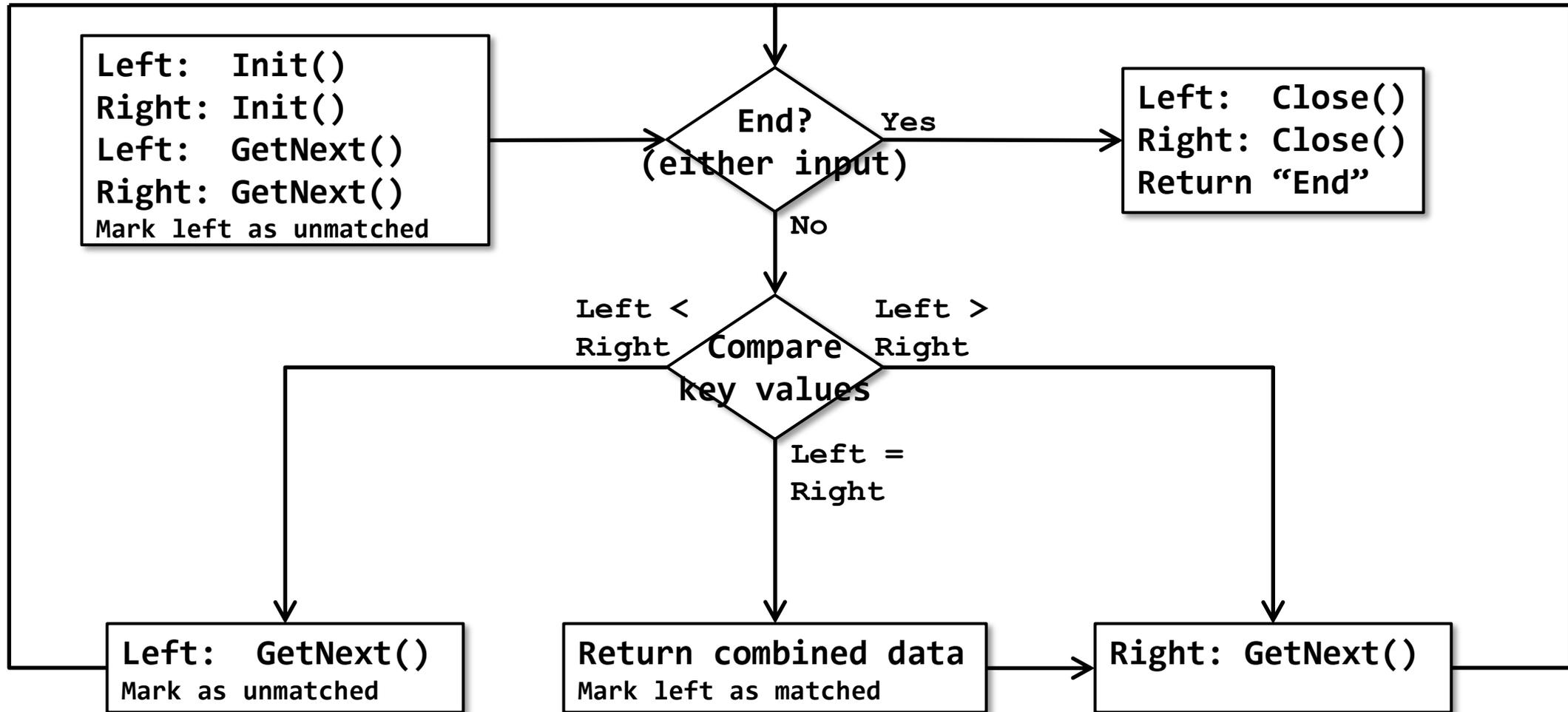
Left-retained rows are out of order for columns from right input

Right-retained rows are out of order for columns from left input

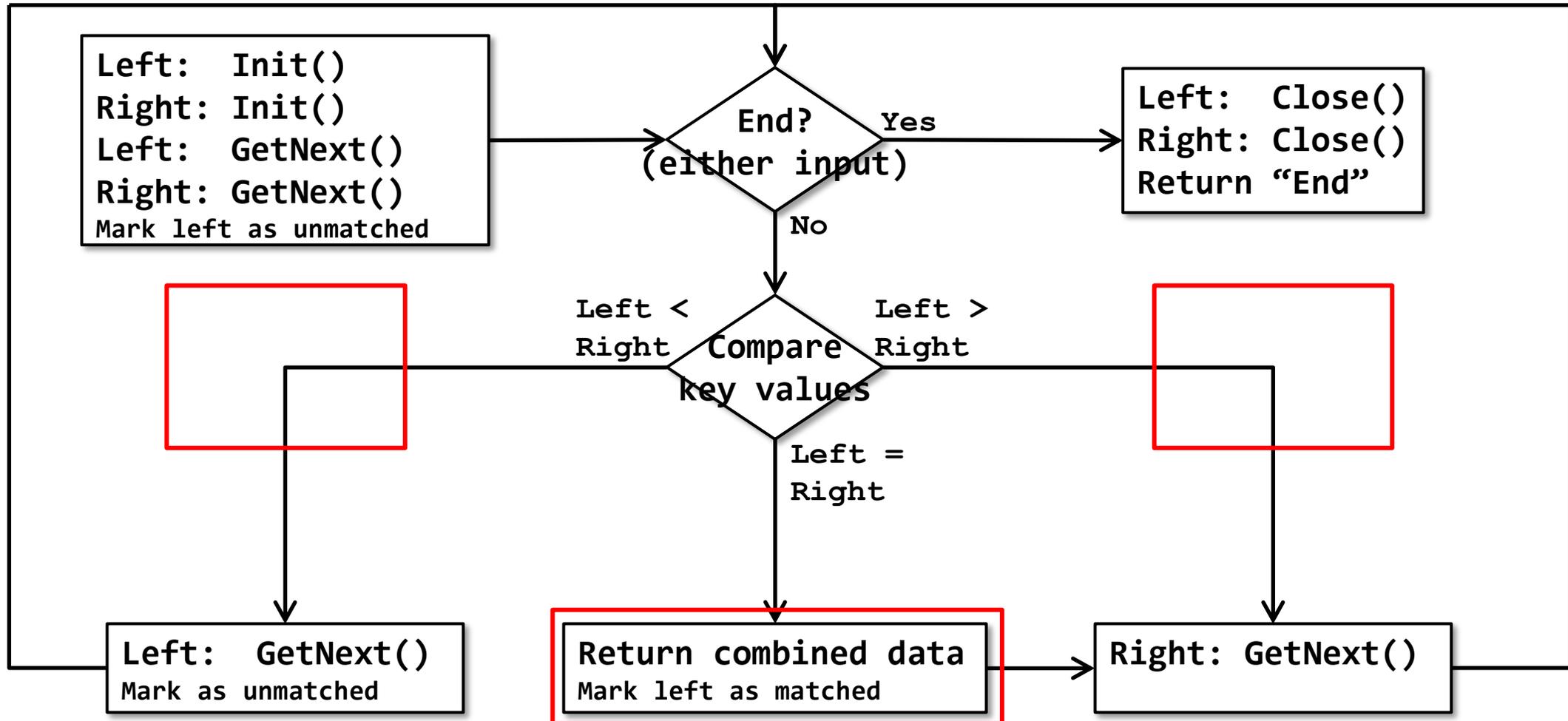


Merge Join  
(Inner Join)

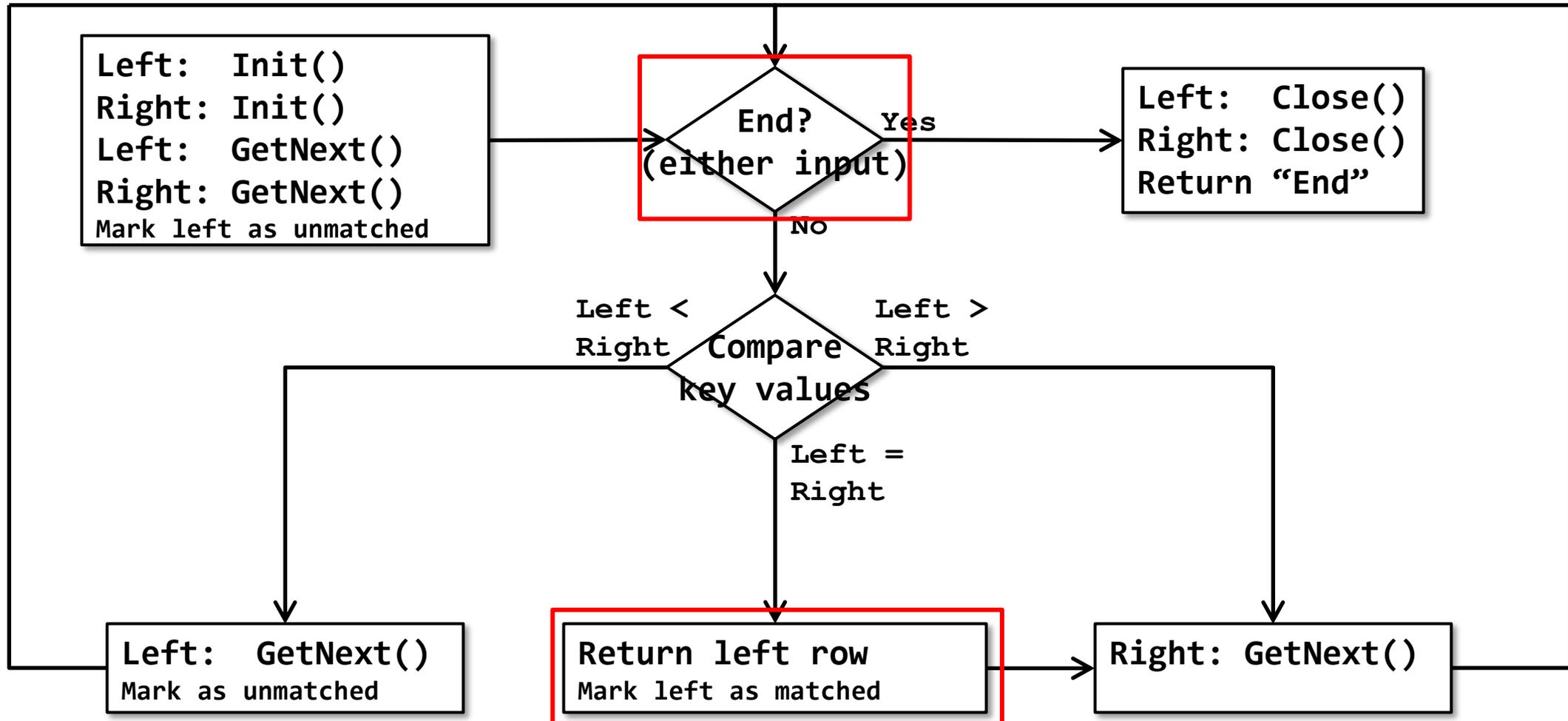
# Merge Join (inner join, one to many)



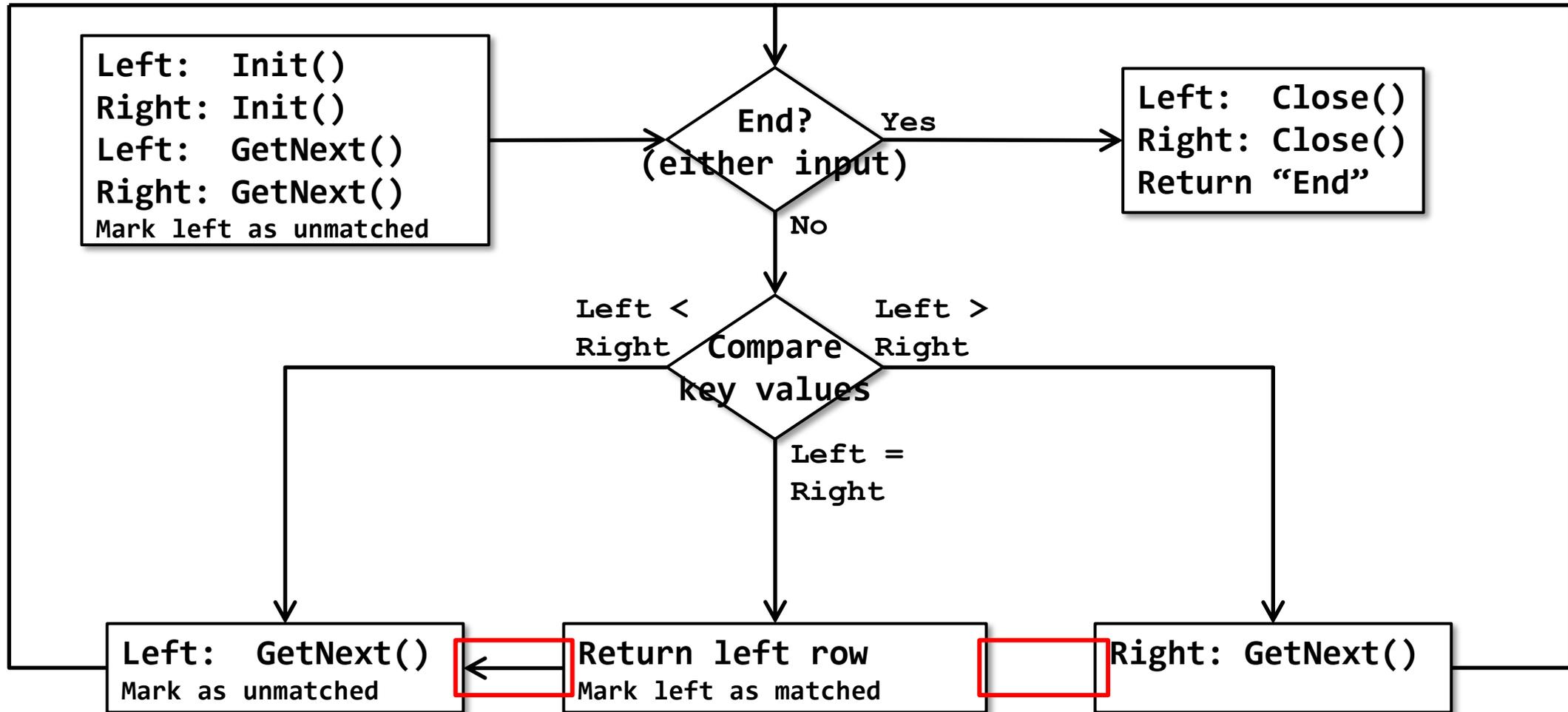
# Merge Join (inner to left semi)



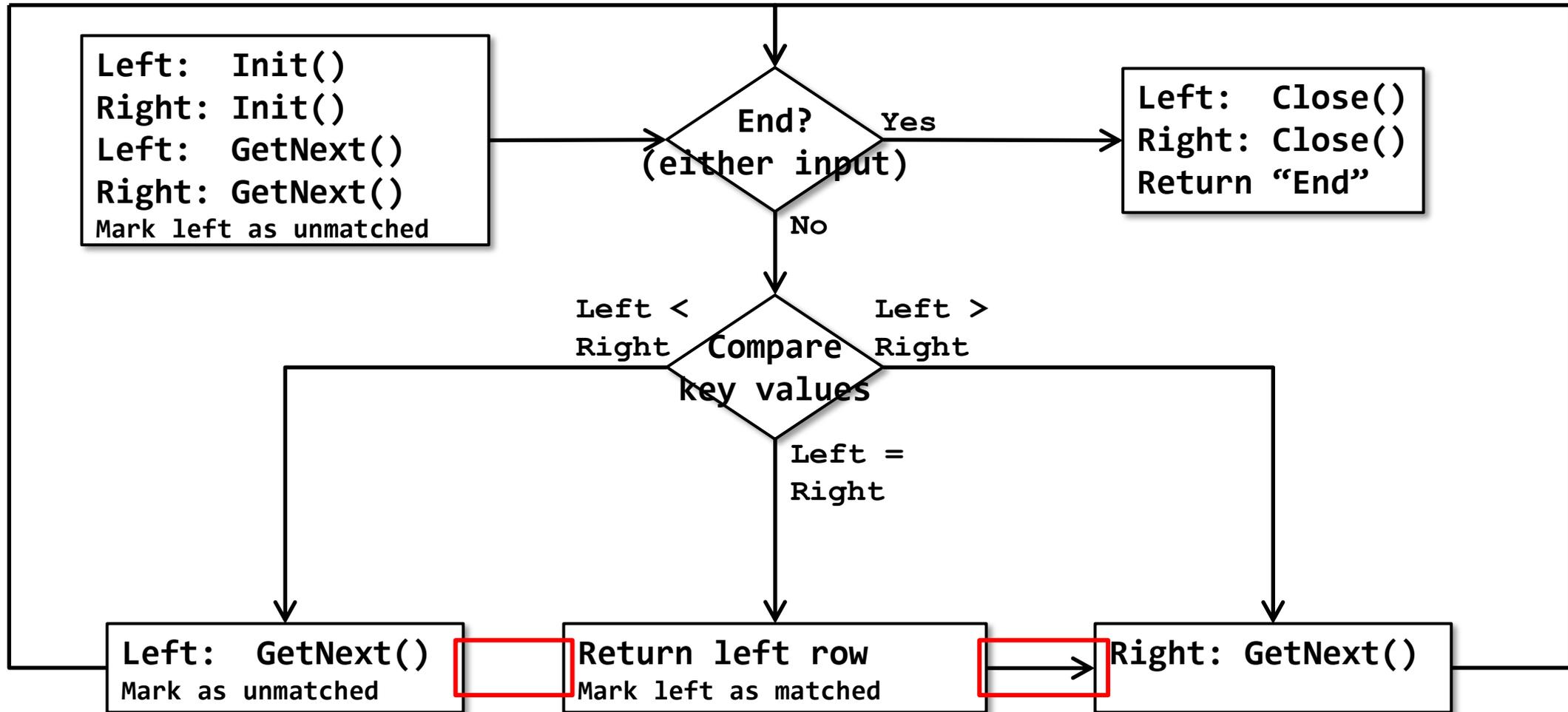
# Merge Join (inner to left semi)



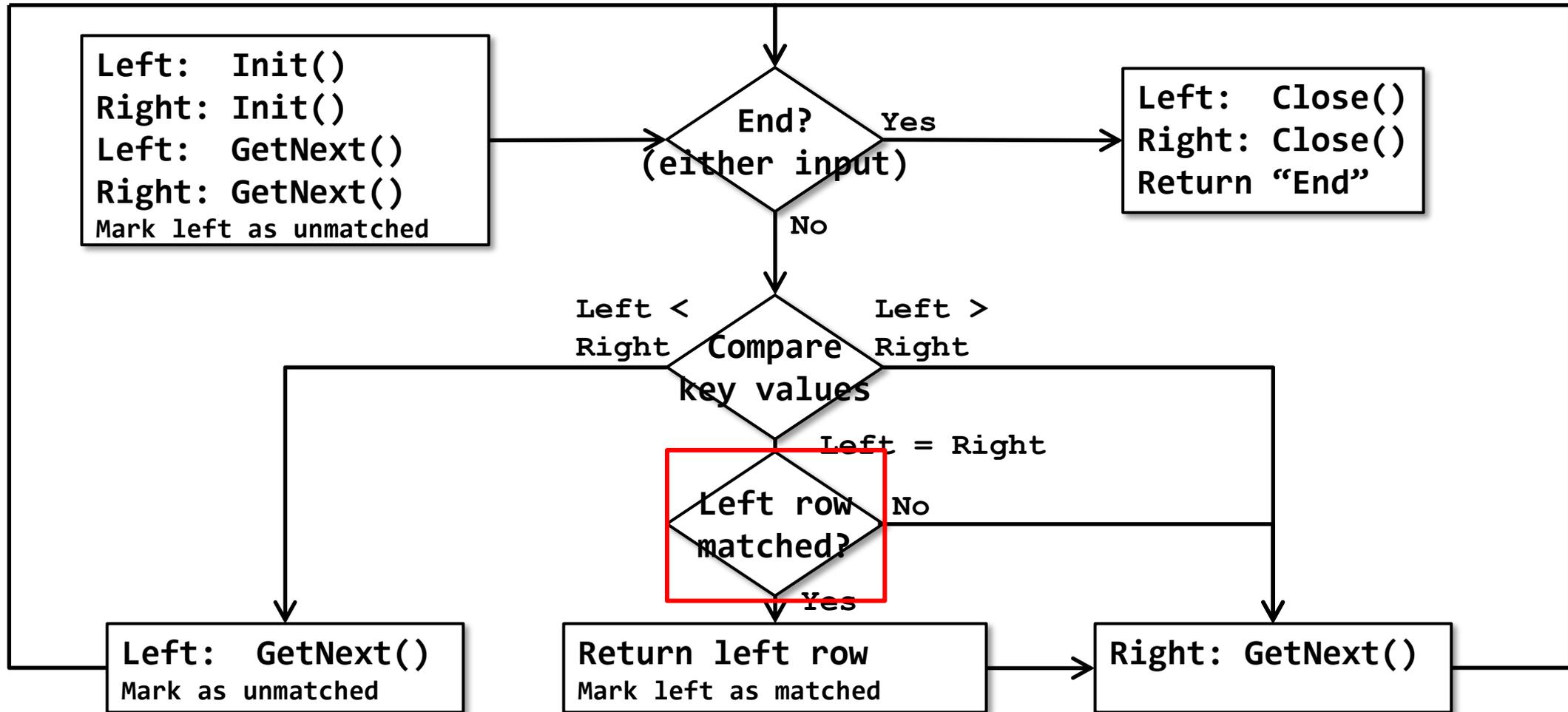
# Merge Join (inner to left semi ??????)



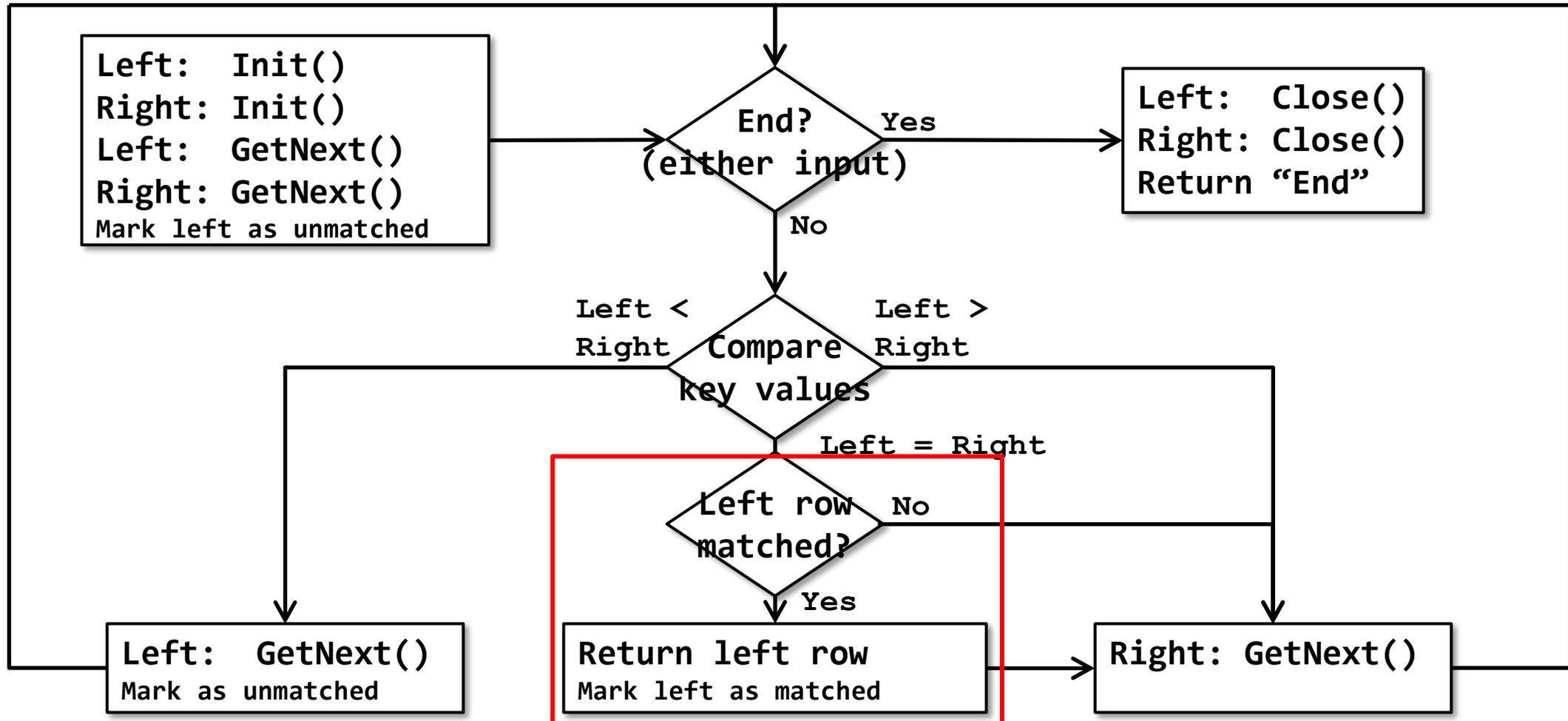
# Merge Join (inner to left semi)



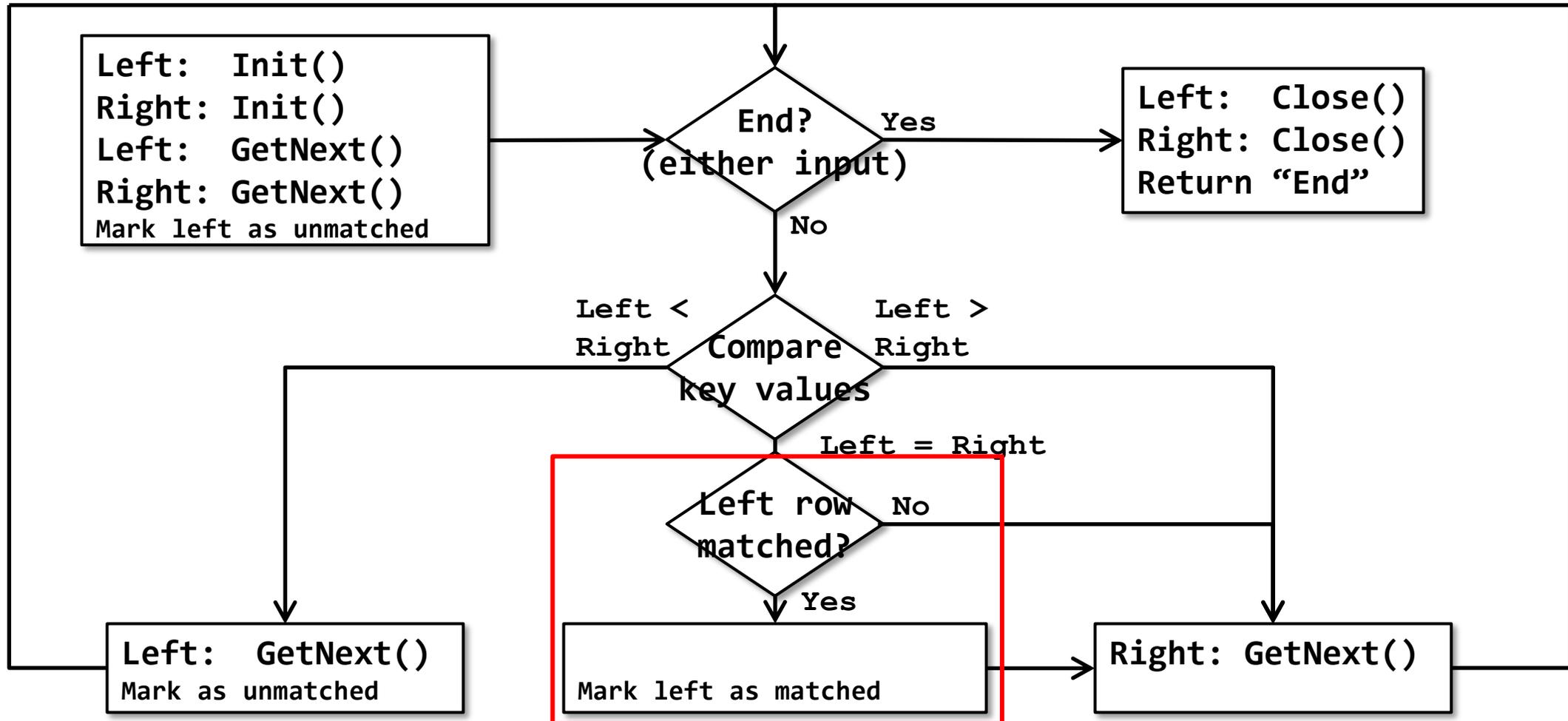
# Merge Join (left semi join, one to many)



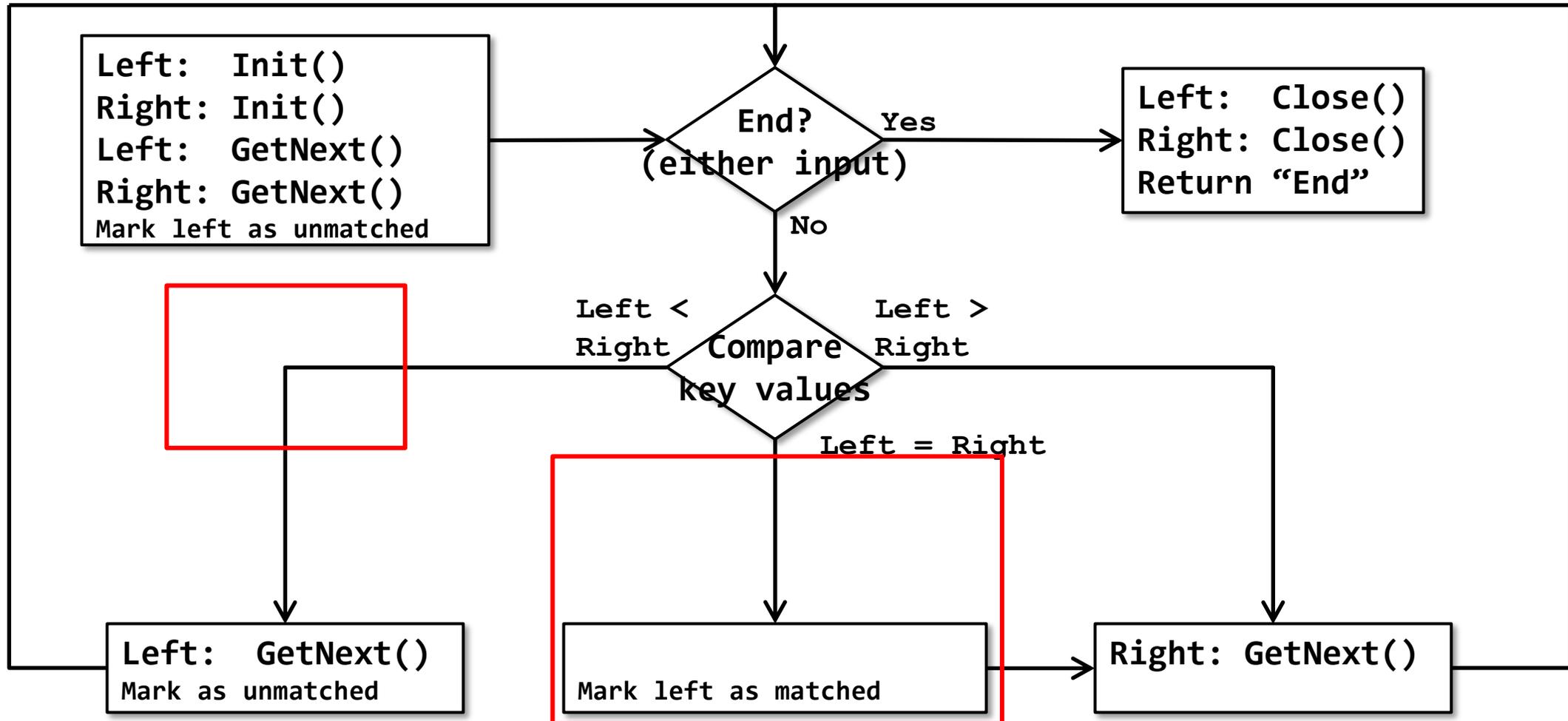
# Merge Join (left semi to left anti semi)



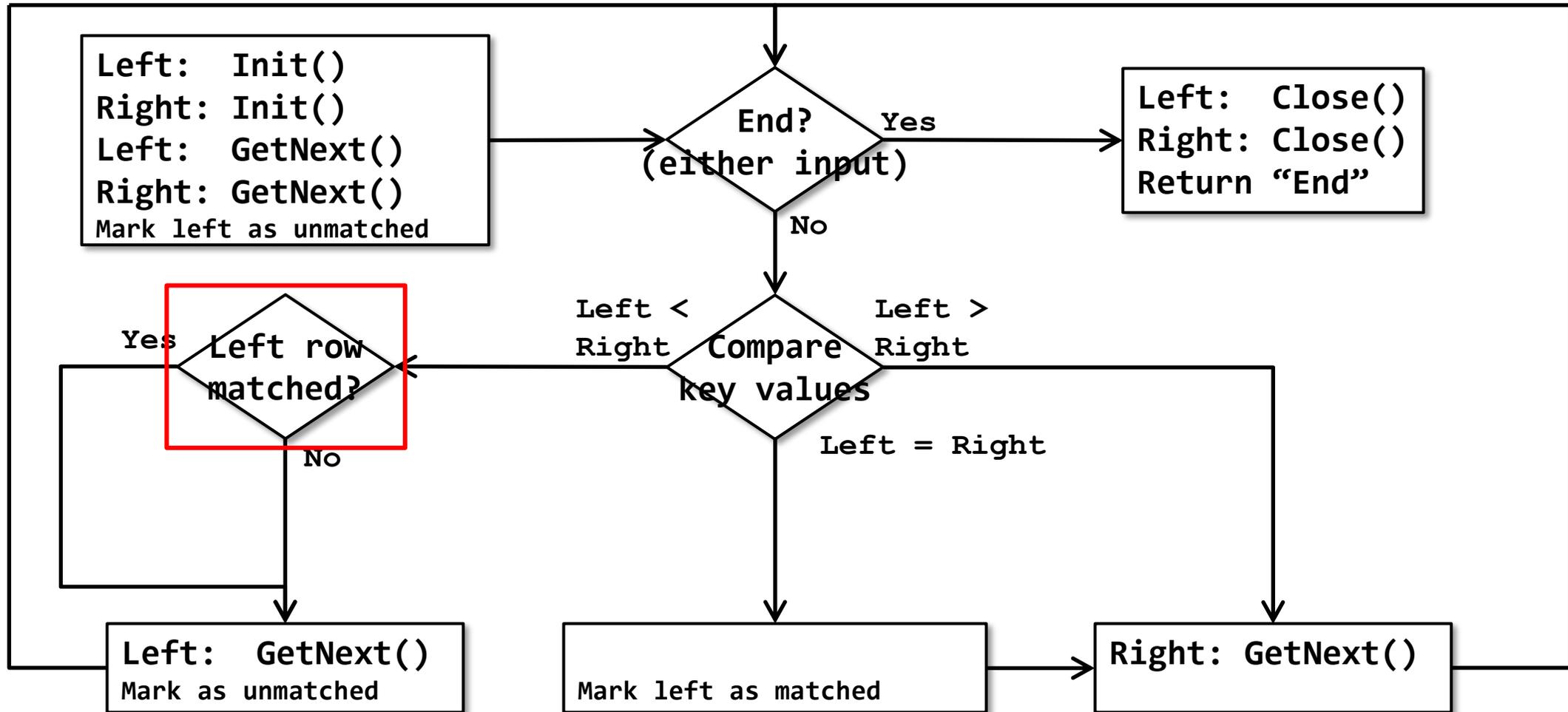
# Merge Join (left semi to left anti semi)



# Merge Join (left semi to left anti semi)

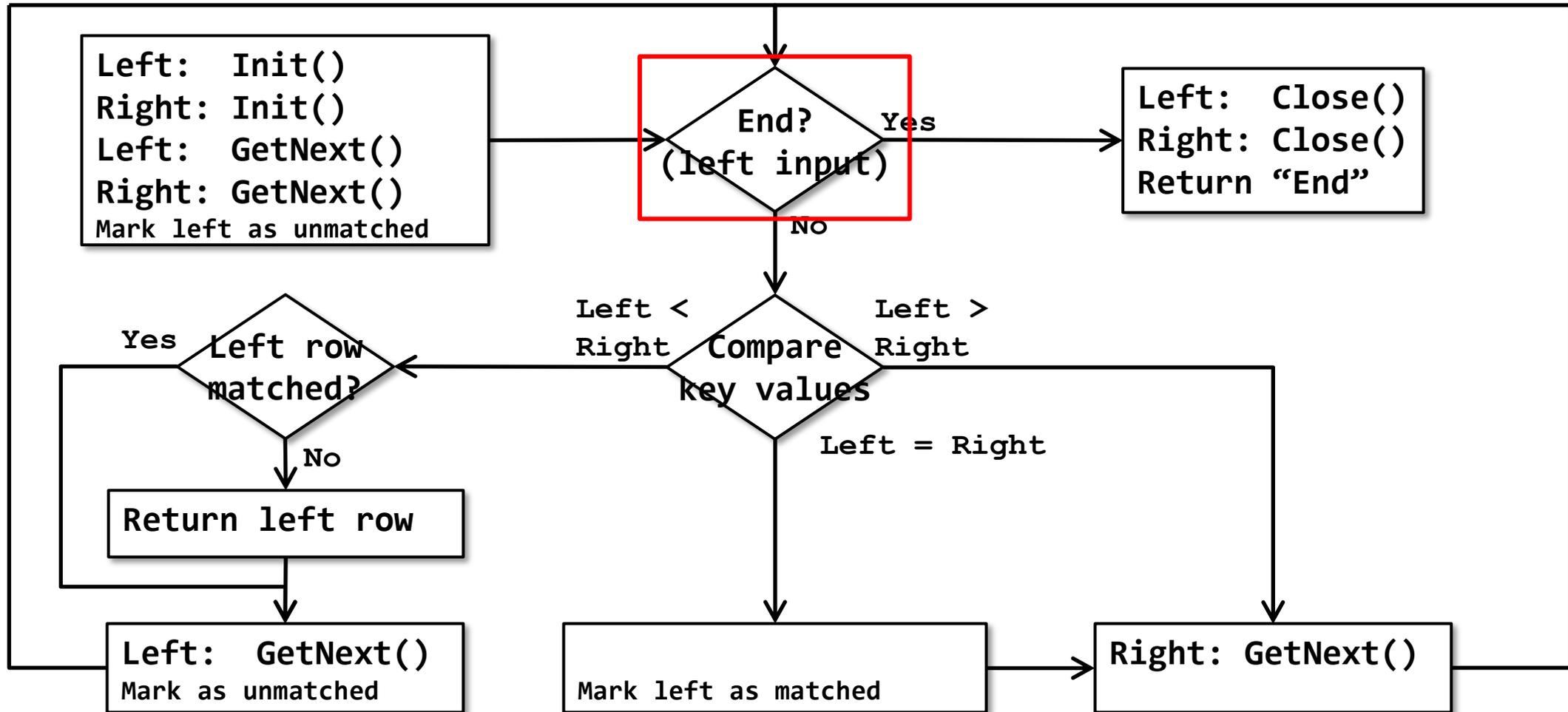


# Merge Join (left semi to left anti semi)

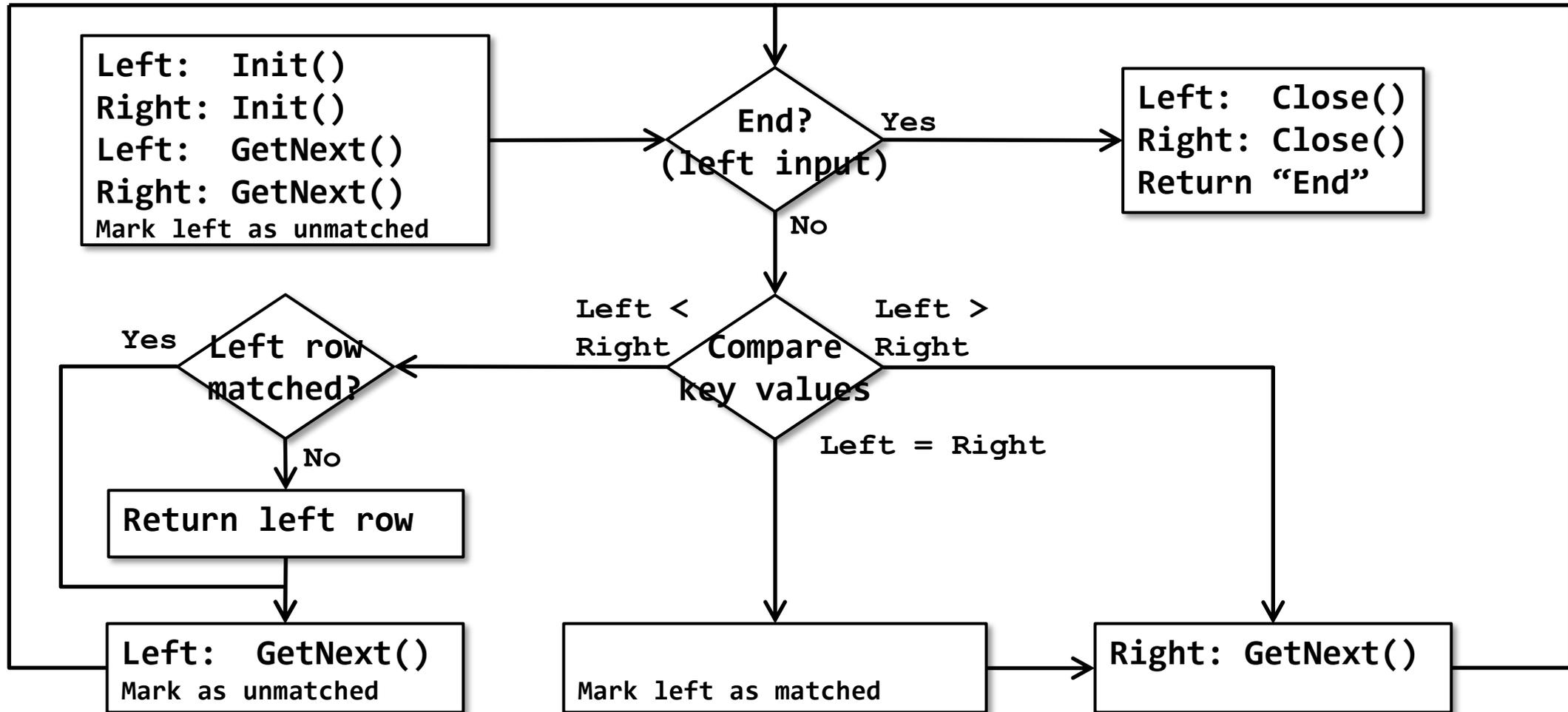




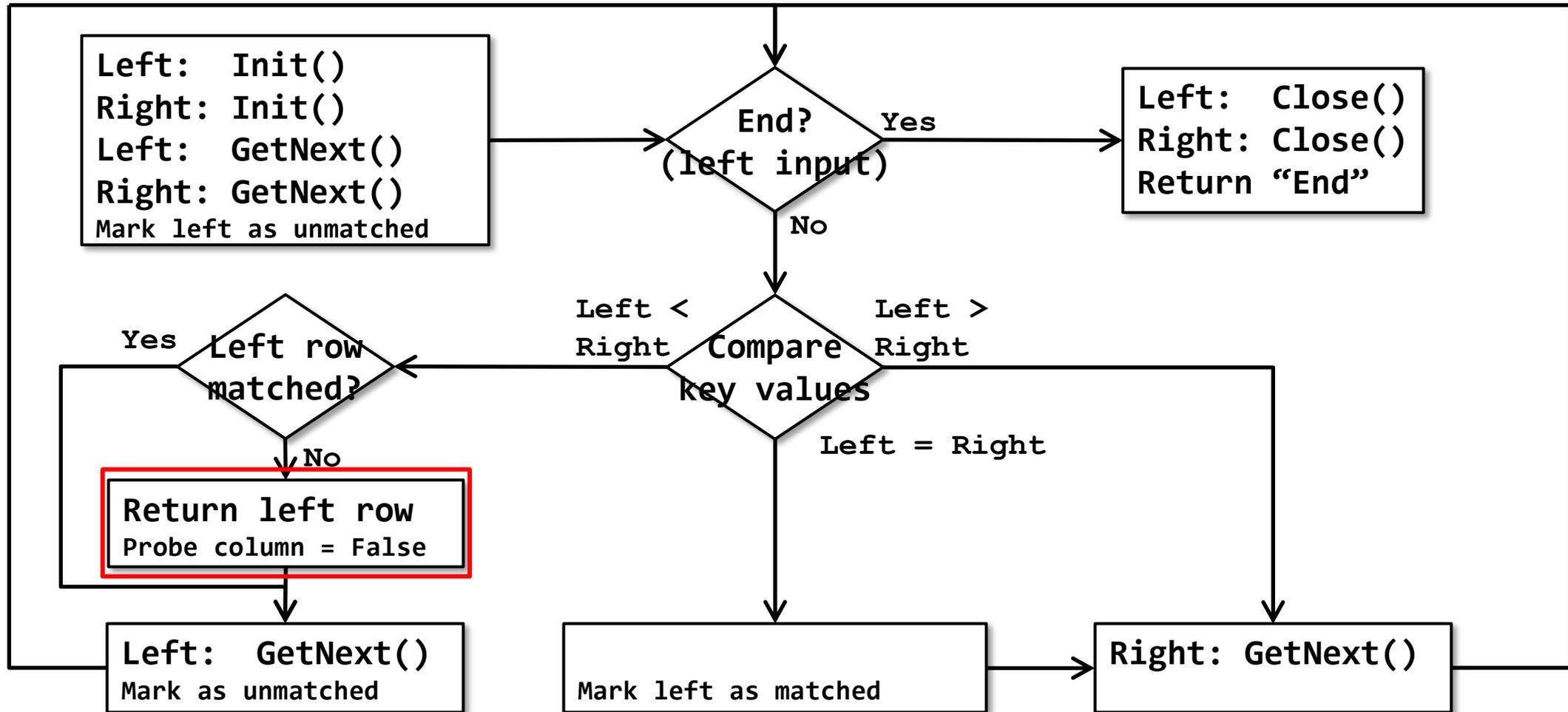
# Merge Join (left anti semi join, one to many)



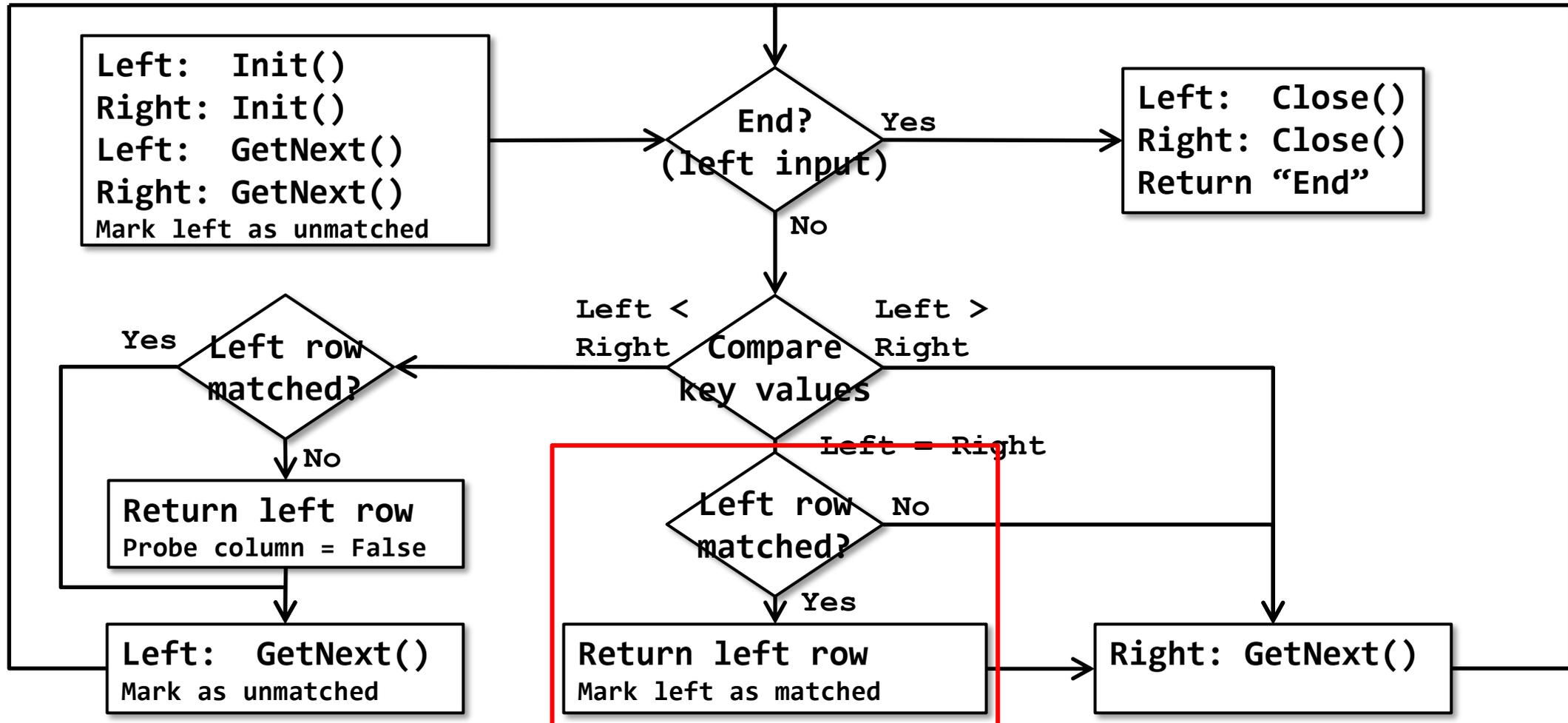
# Merge Join (left anti semi to probed left semi)



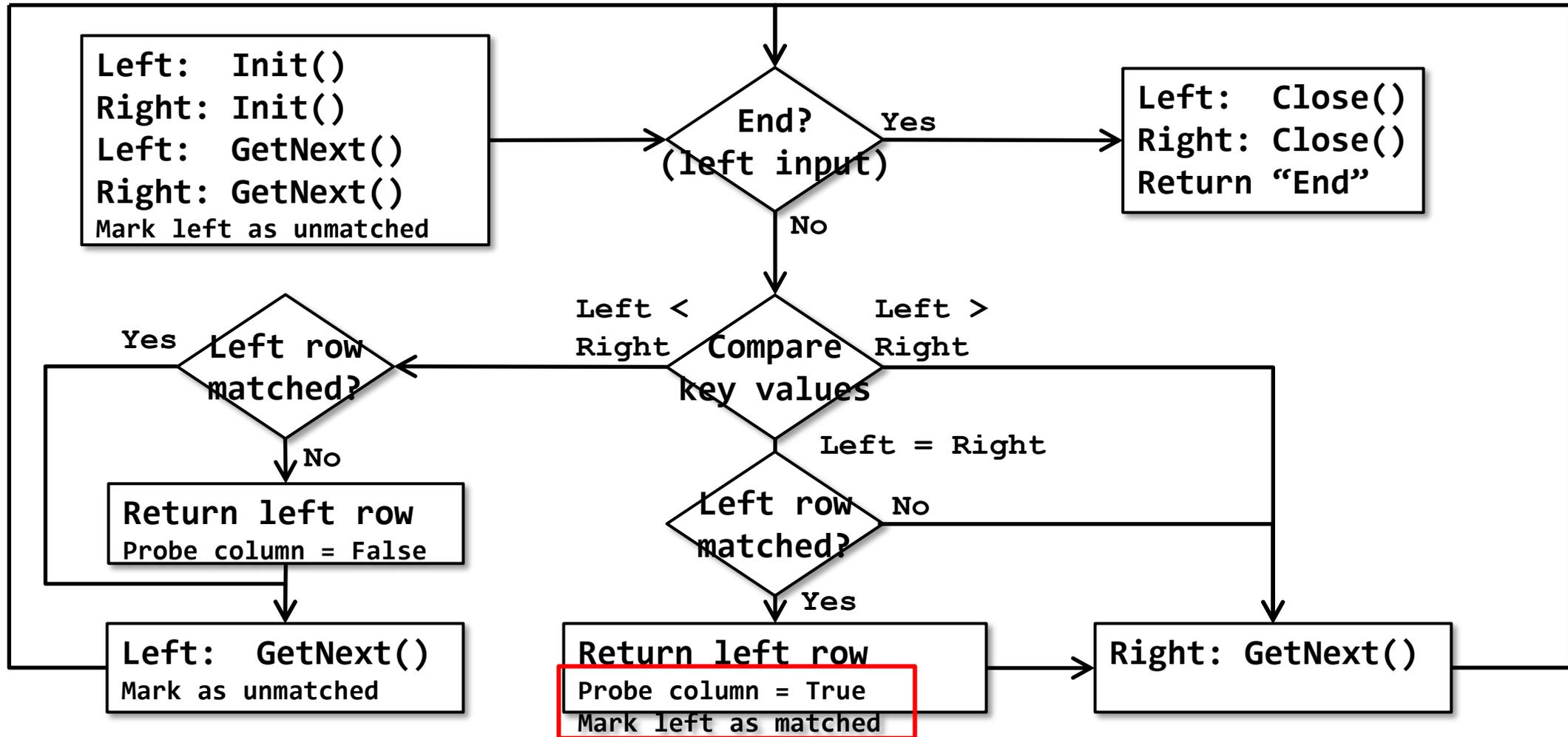
# Merge Join (left anti semi to probed left semi)



# Merge Join (left anti semi to probed left semi)



# Merge Join (probed left semi join, one to many)



# Merge Join

## Merge Join

Retains order of both inputs for Inner Join

Retains order of left input for Left Outer Join

Retains order of left input for Right Outer Join

Retains order of none of the inputs for Full Outer Join

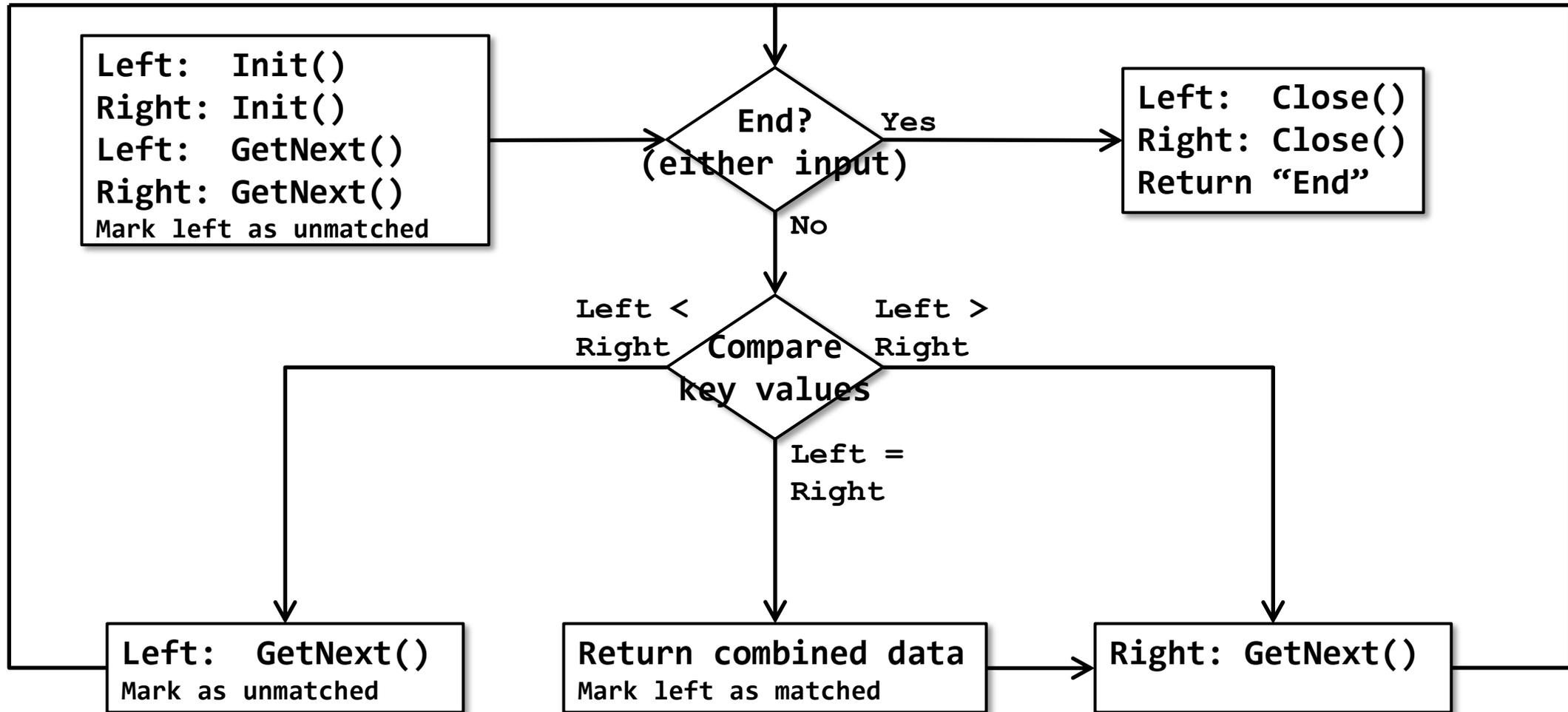
Retains order of left input for all Left Semi Join variants

Right input not in output

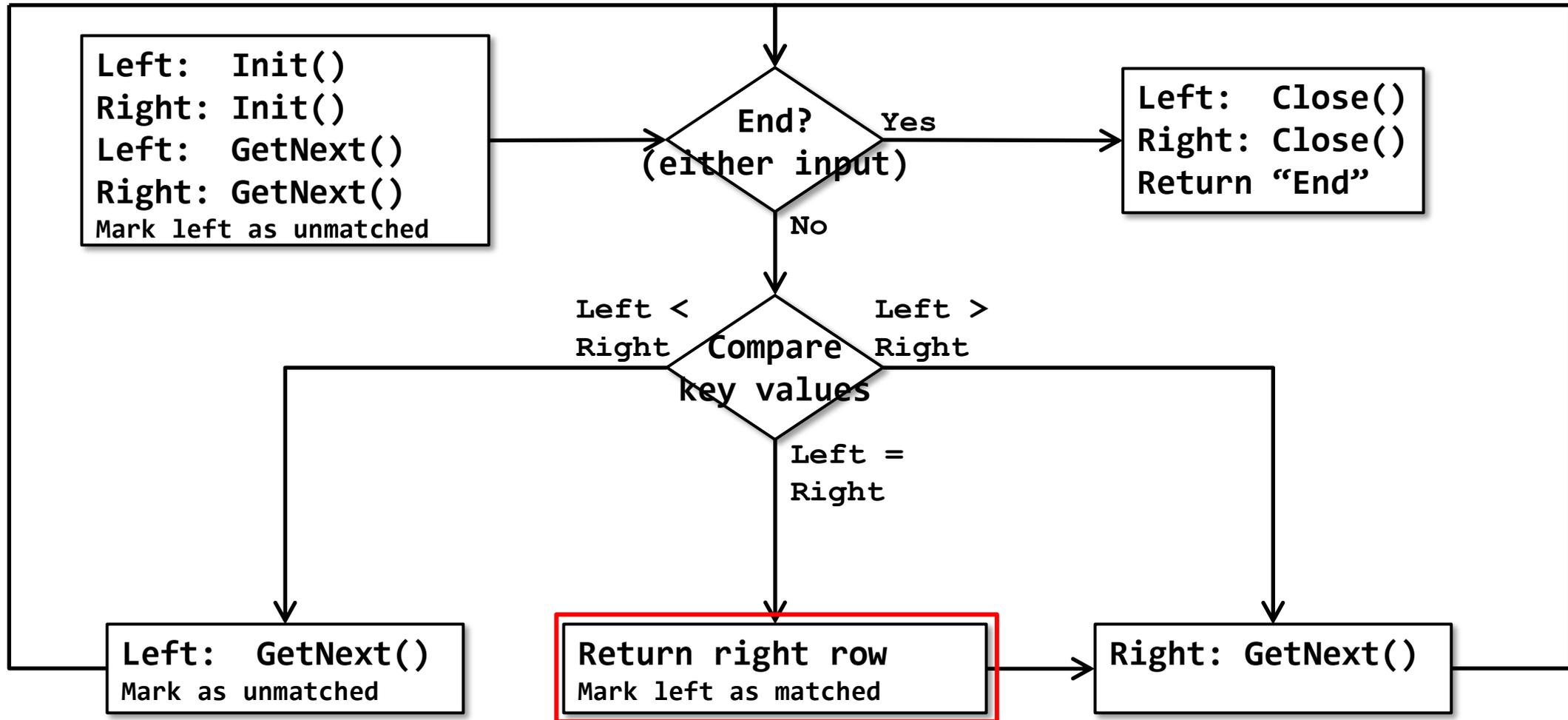


Merge Join  
(Inner Join)

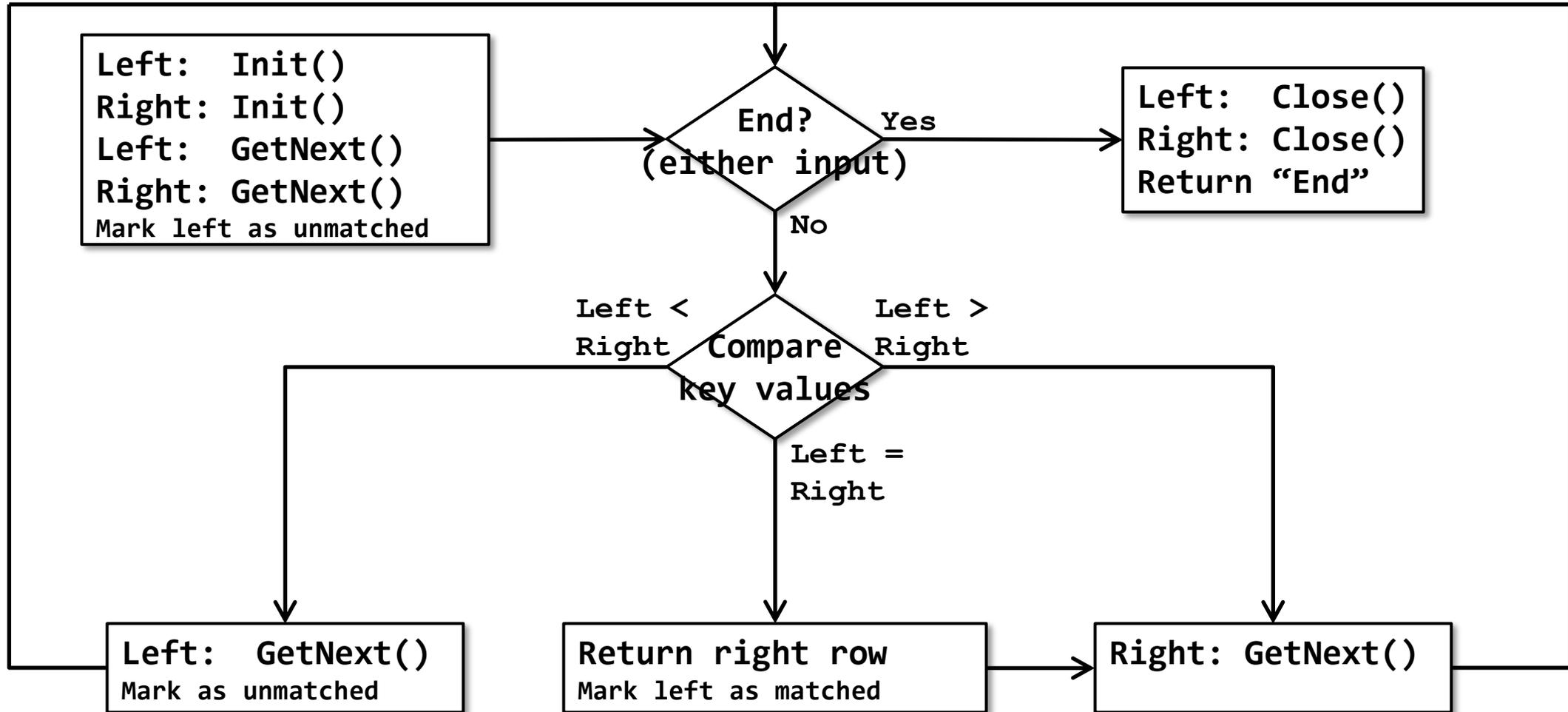
# Merge Join (inner to right semi)



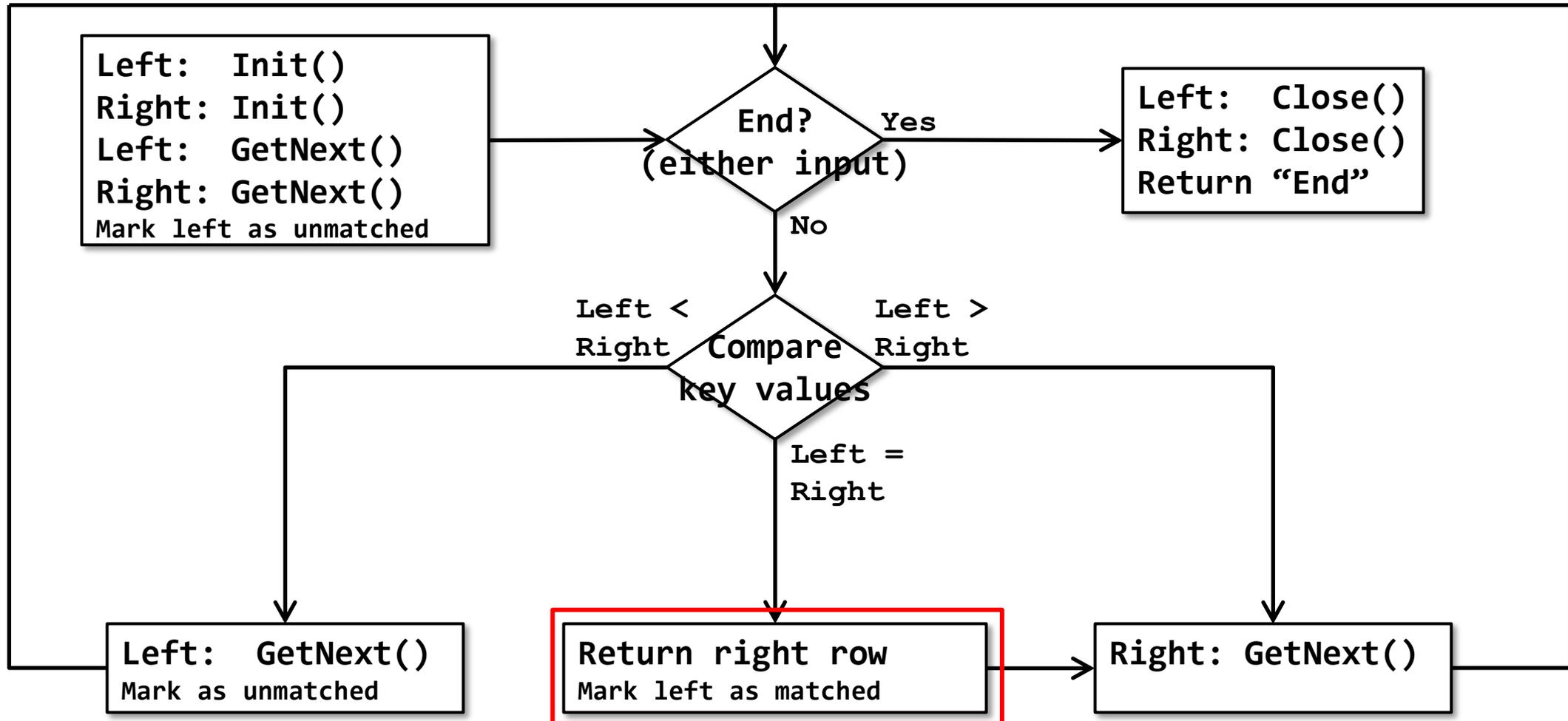
# Merge Join (inner to right semi)



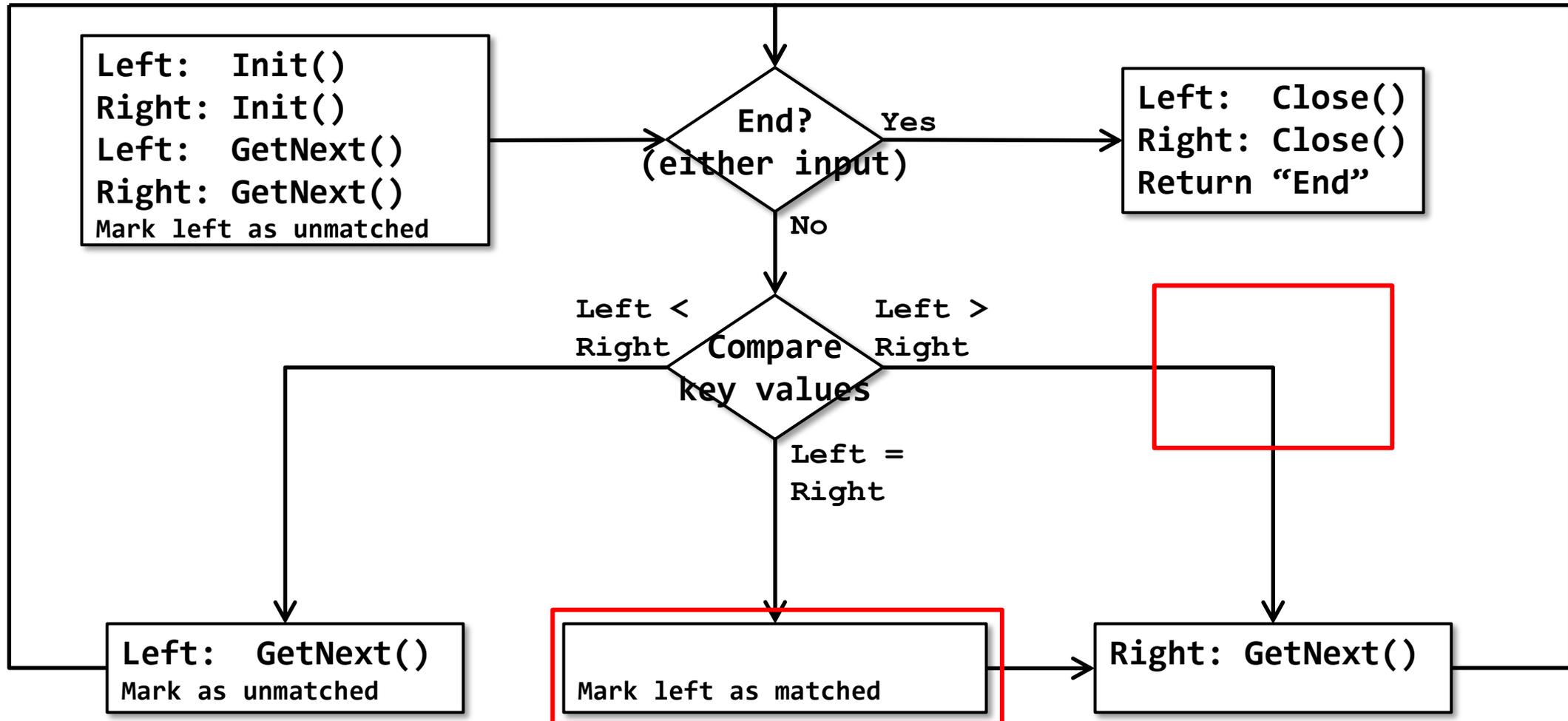
# Merge Join (right semi join, one to many)



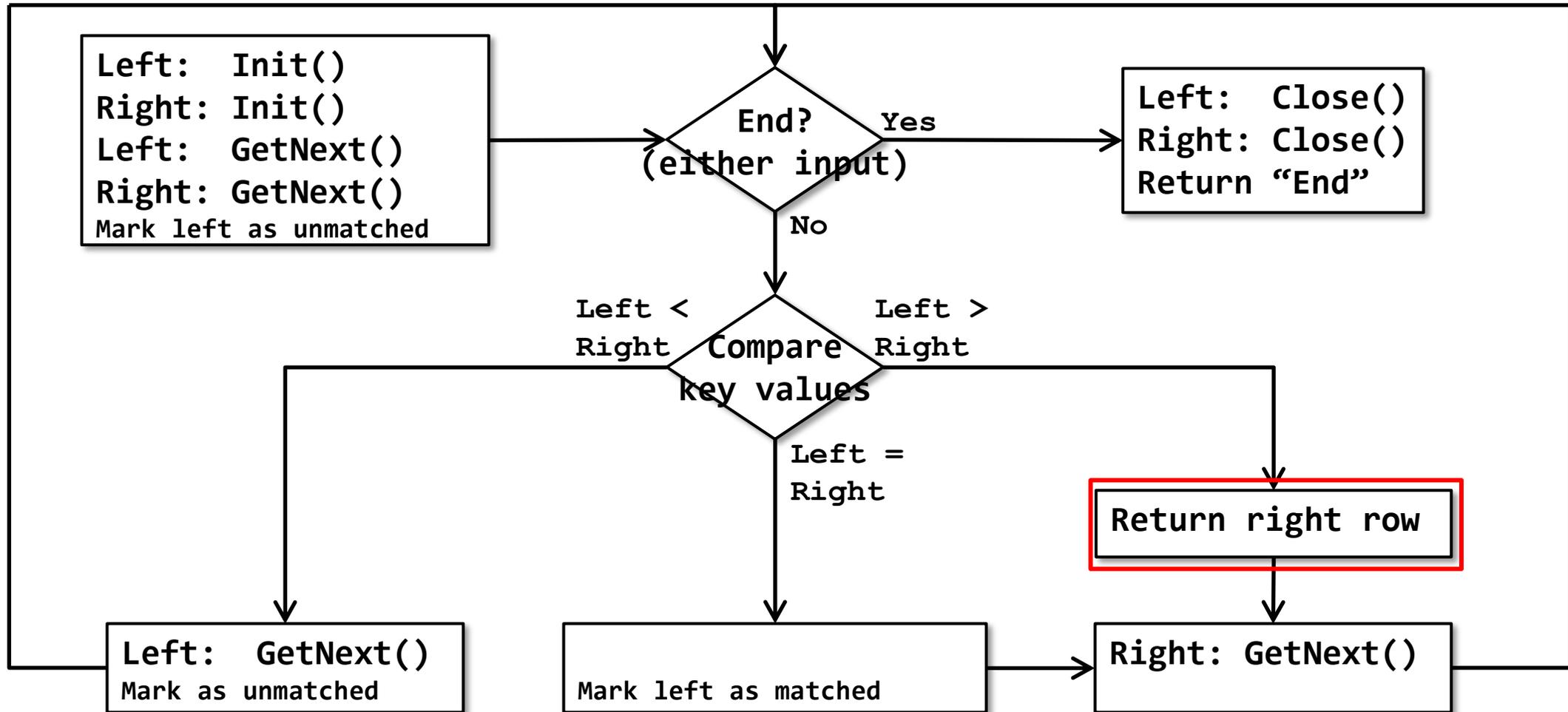
# Merge Join (right semi to right anti semi)



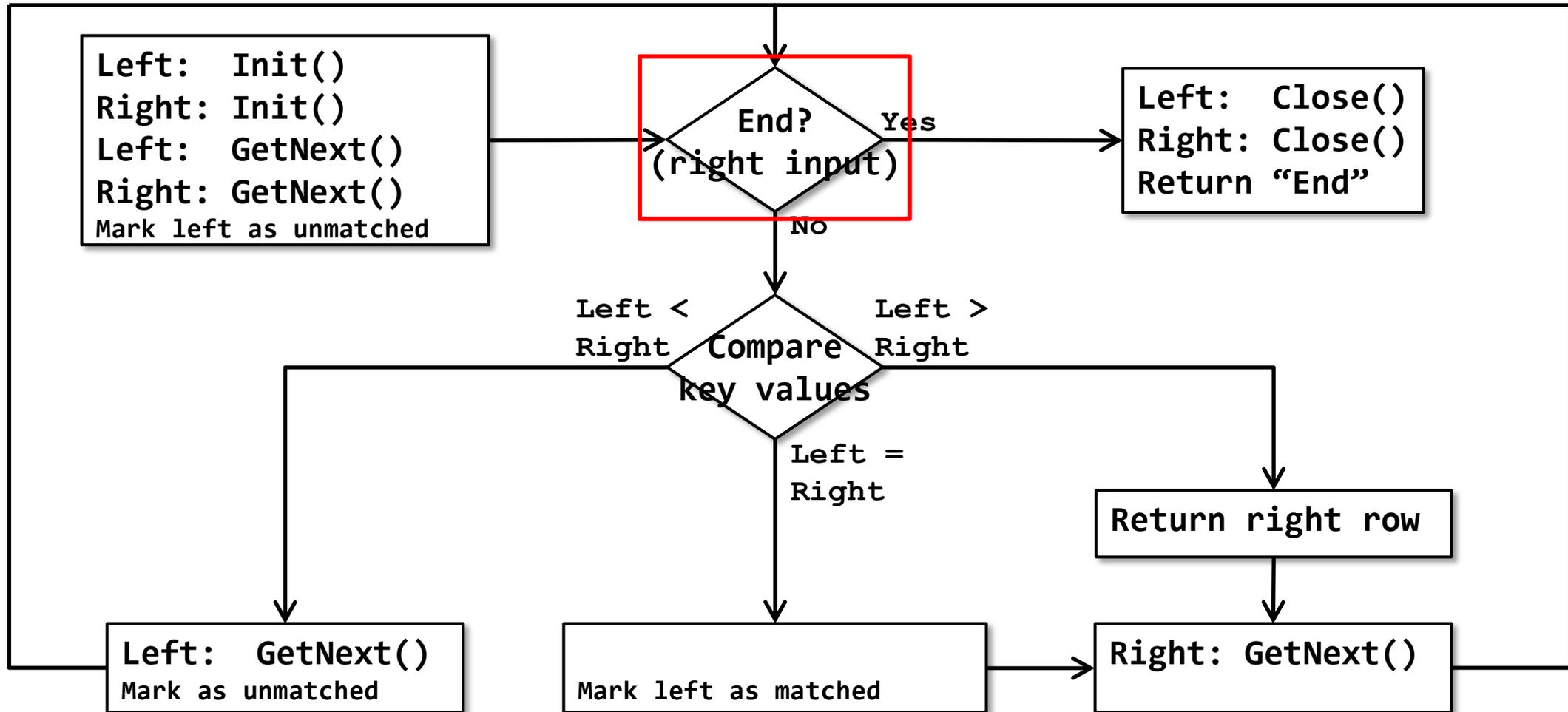
# Merge Join (right semi to right anti semi)



# Merge Join (right anti semi join, one to many)



# Merge Join (right anti semi join, one to many)



# Merge Join

## Merge Join

Retains order of both inputs for Inner Join

Retains order of left input for Left Outer Join

Retains order of left input for Right Outer Join

Retains order of none of the inputs for Full Outer Join

Retains order of left input for all Left Semi Join variants

Retains order of right input for all Right Semi Join variants

Left input not in output



Merge Join  
(Inner Join)

# Merge Join

## Merge Join

Logical operations supported

Inner Join

Left / Right / Full Outer Join

Left / Right Semi Join

Left / Right Anti Semi Join

Probed Left Semi Join

Concatenation

Union



Merge Join  
(Inner Join)

# Merge Join

## Merge Join

Logical operations supported

Inner Join

Left / Right / Full Outer Join

Left / Right Semi Join

Left / Right Anti Semi Join

Probed Left Semi Join

Concatenation → Similar to UNION ALL in T-SQL

Union



Merge Join  
(Inner Join)

# Merge Join

## Merge Join

Retains order of both inputs for Inner Join

Retains order of left input for Left Outer Join

Retains order of left input for Right Outer Join

Retains order of none of the inputs for Full Outer Join

Retains order of left input for all Left Semi Join variants

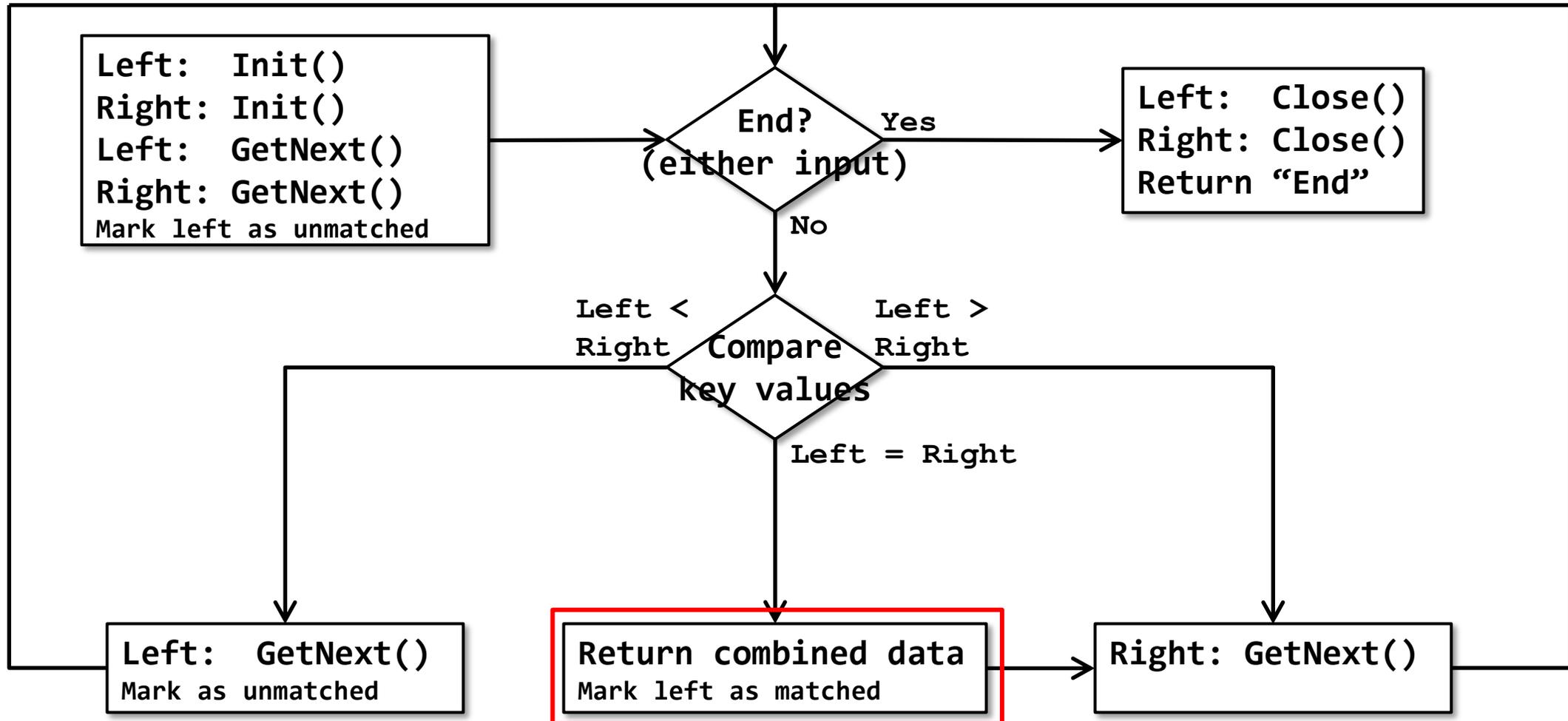
Retains order of right input for all Right Semi Join variants

Retains order of both inputs for Concatenation

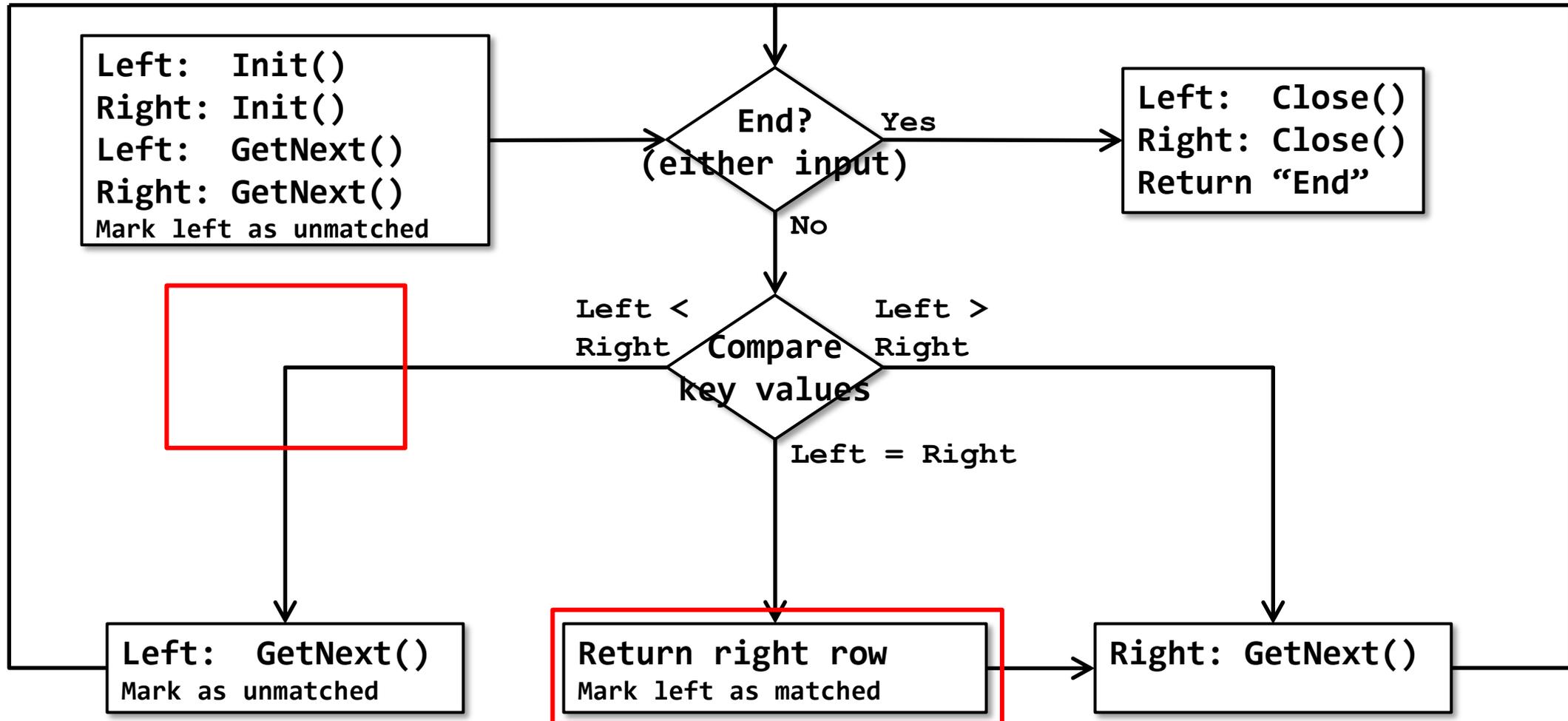


Merge Join  
(Inner Join)

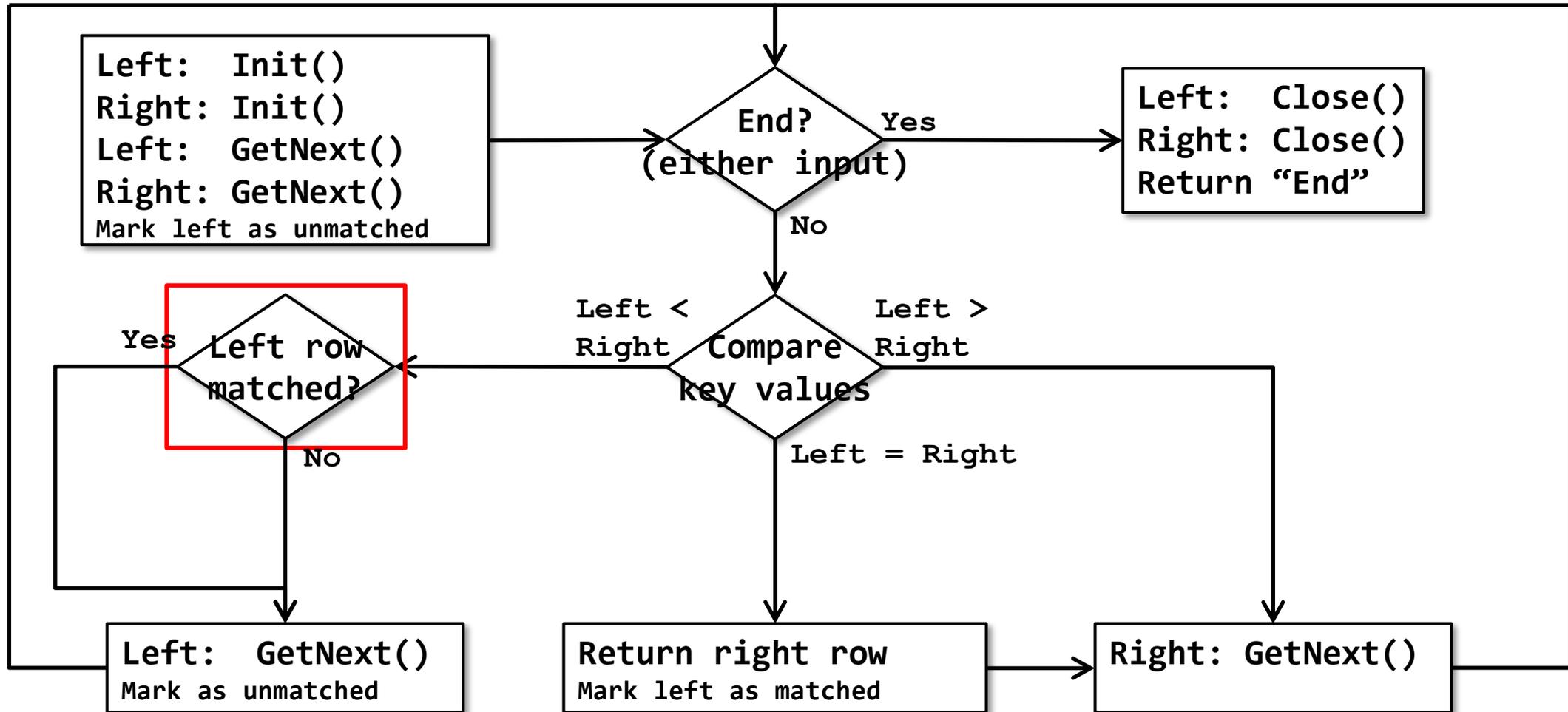
# Merge Join (inner join to concatenation)



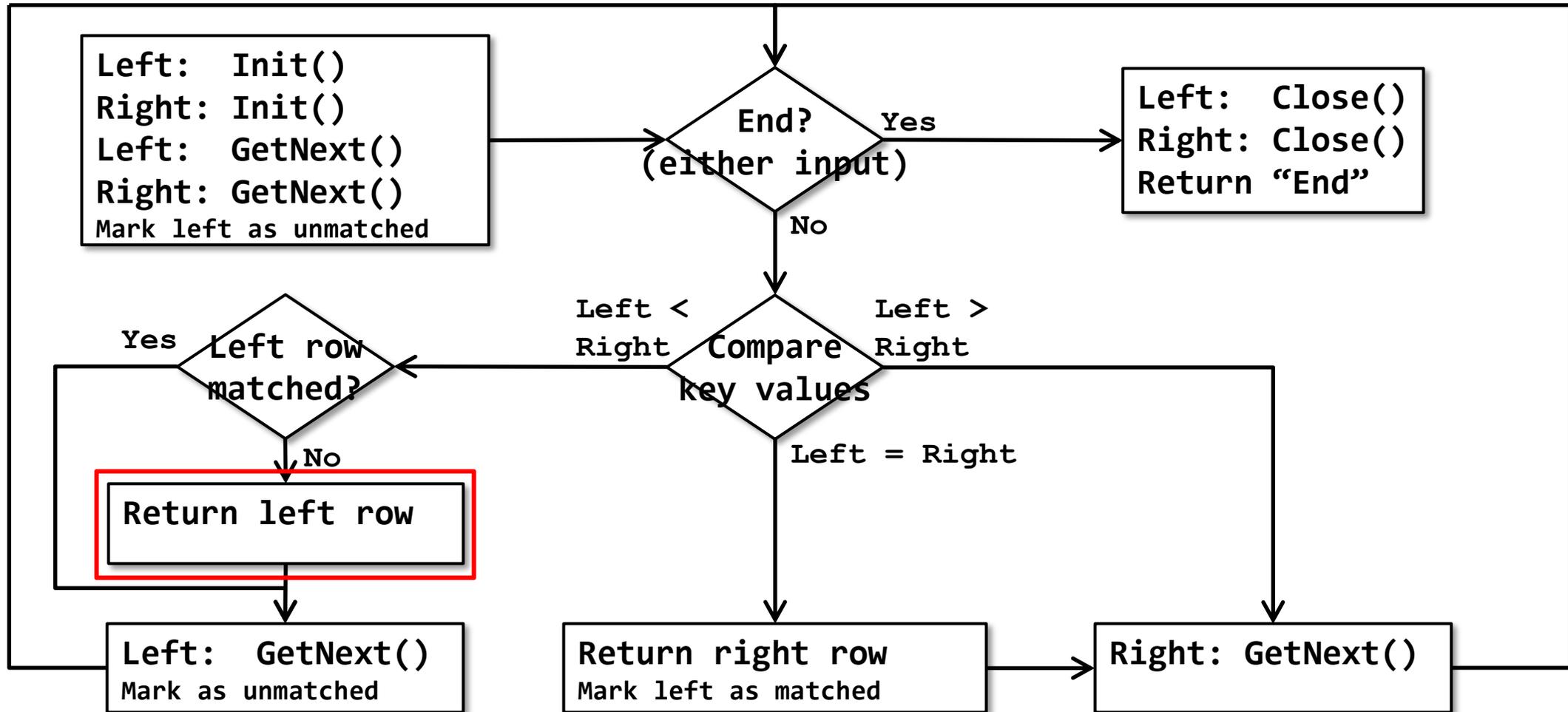
# Merge Join (inner join to concatenation)



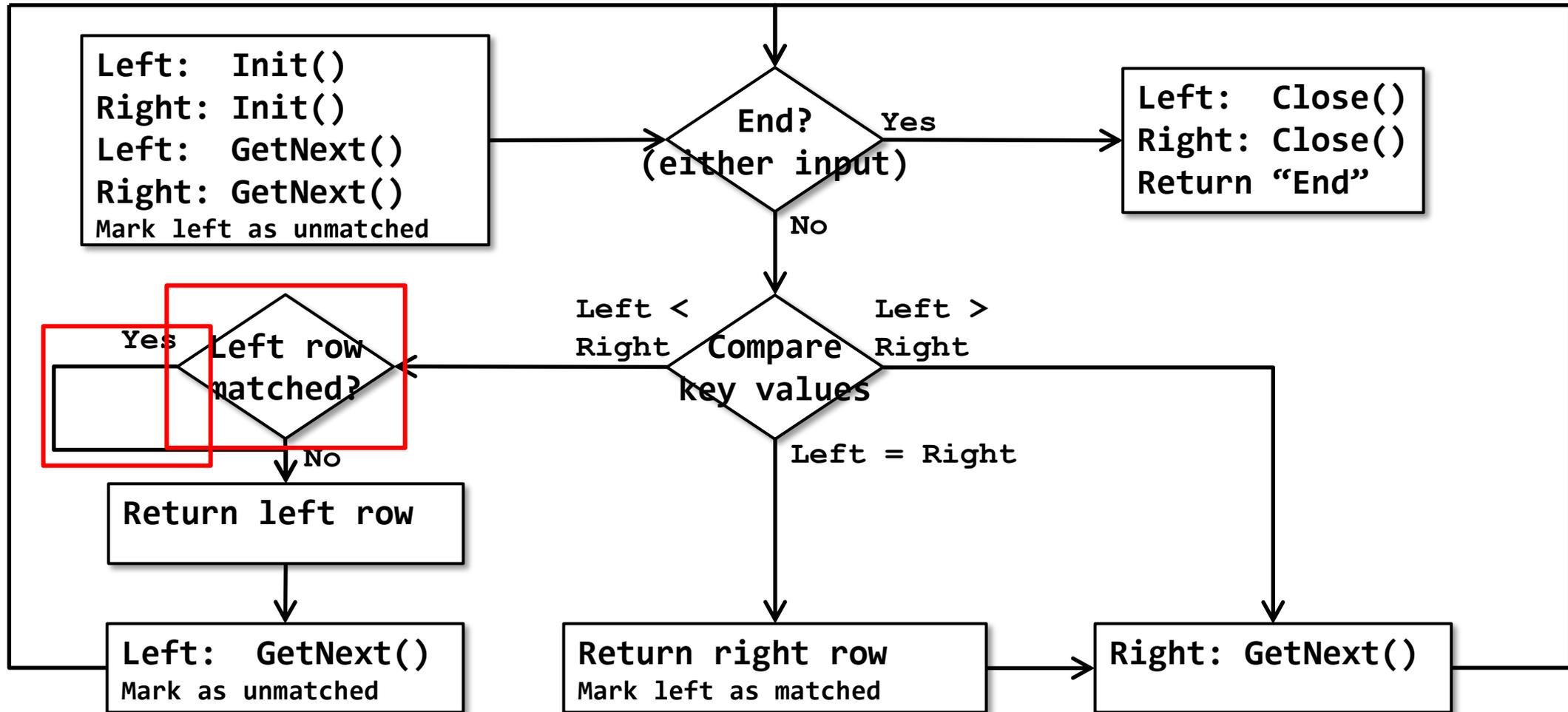
# Merge Join (inner join to concatenation)



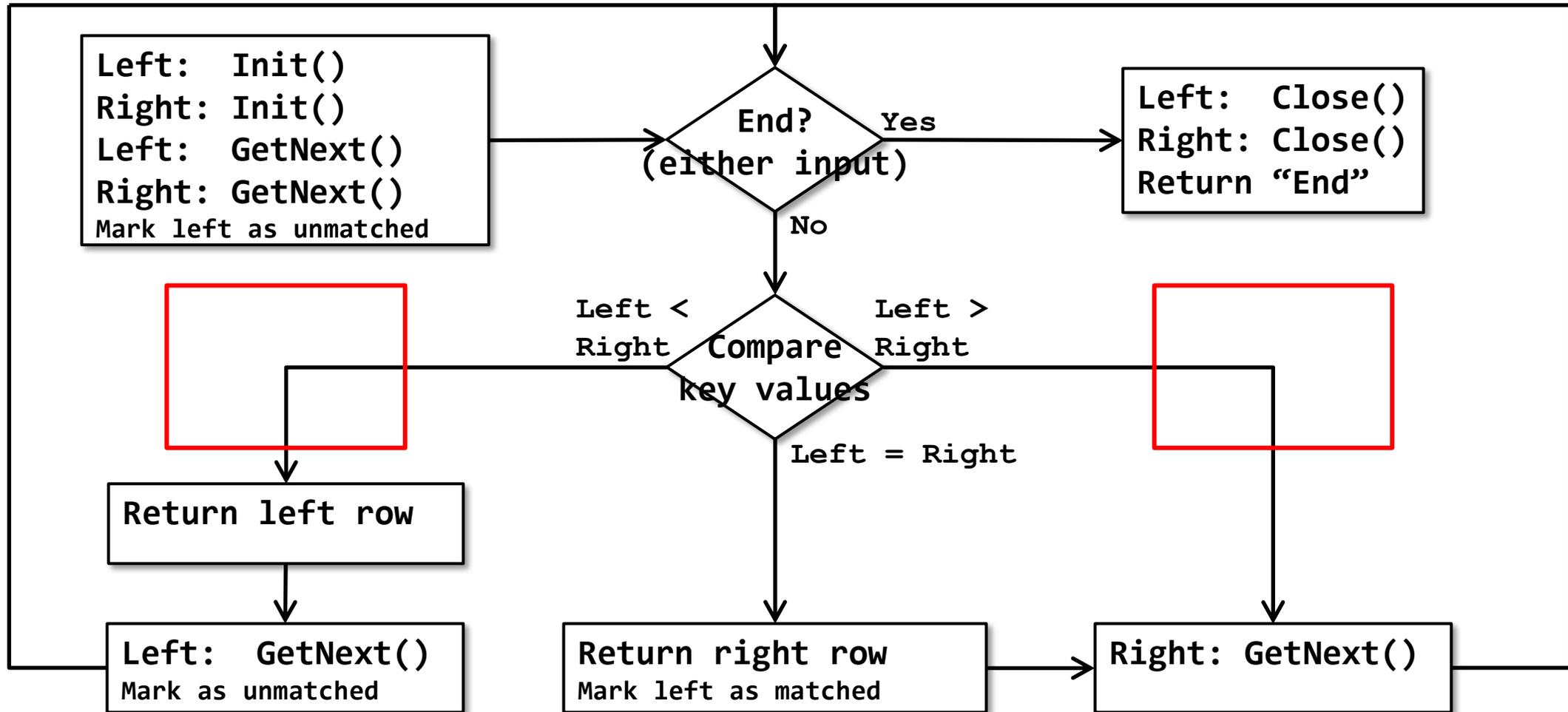
# Merge Join (inner join to concatenation)



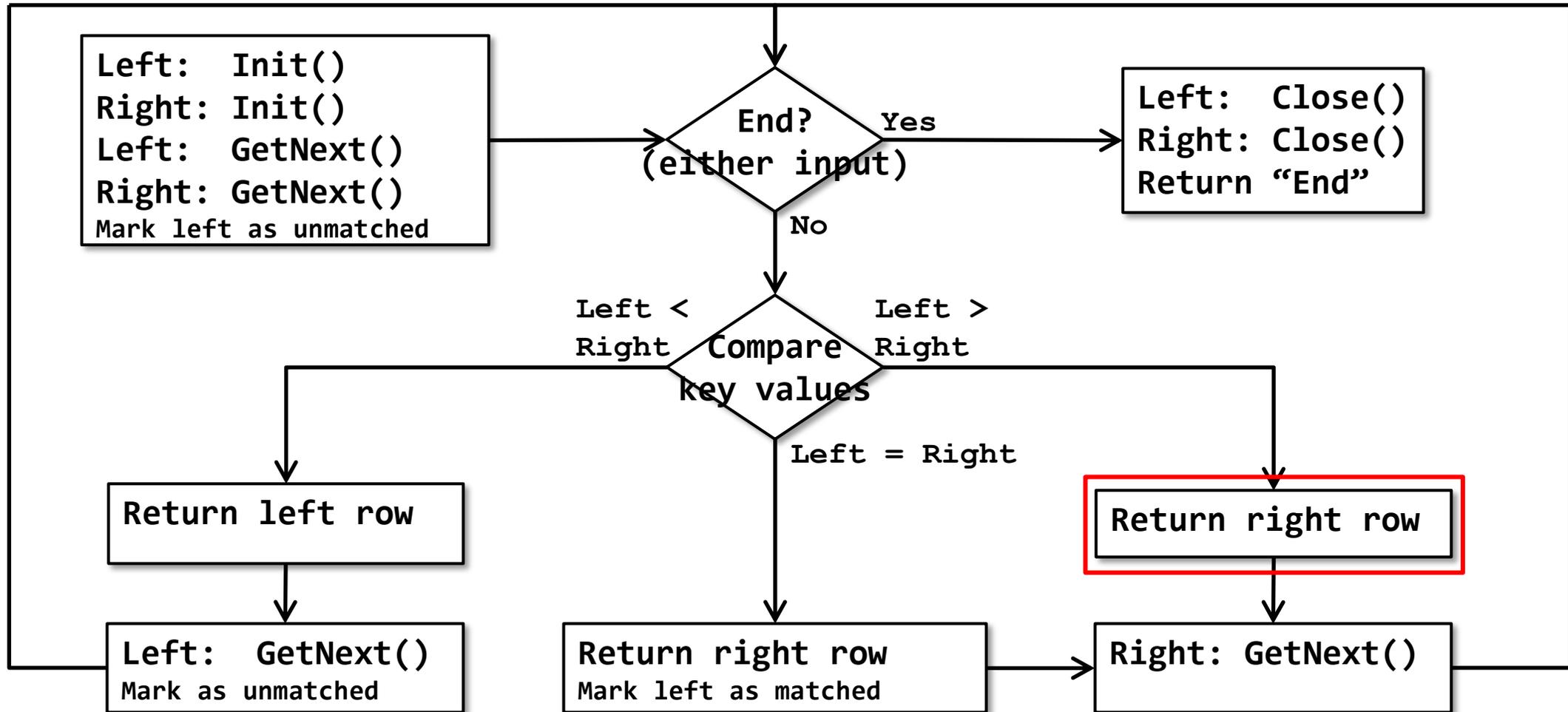
# Merge Join (inner join to concatenation)



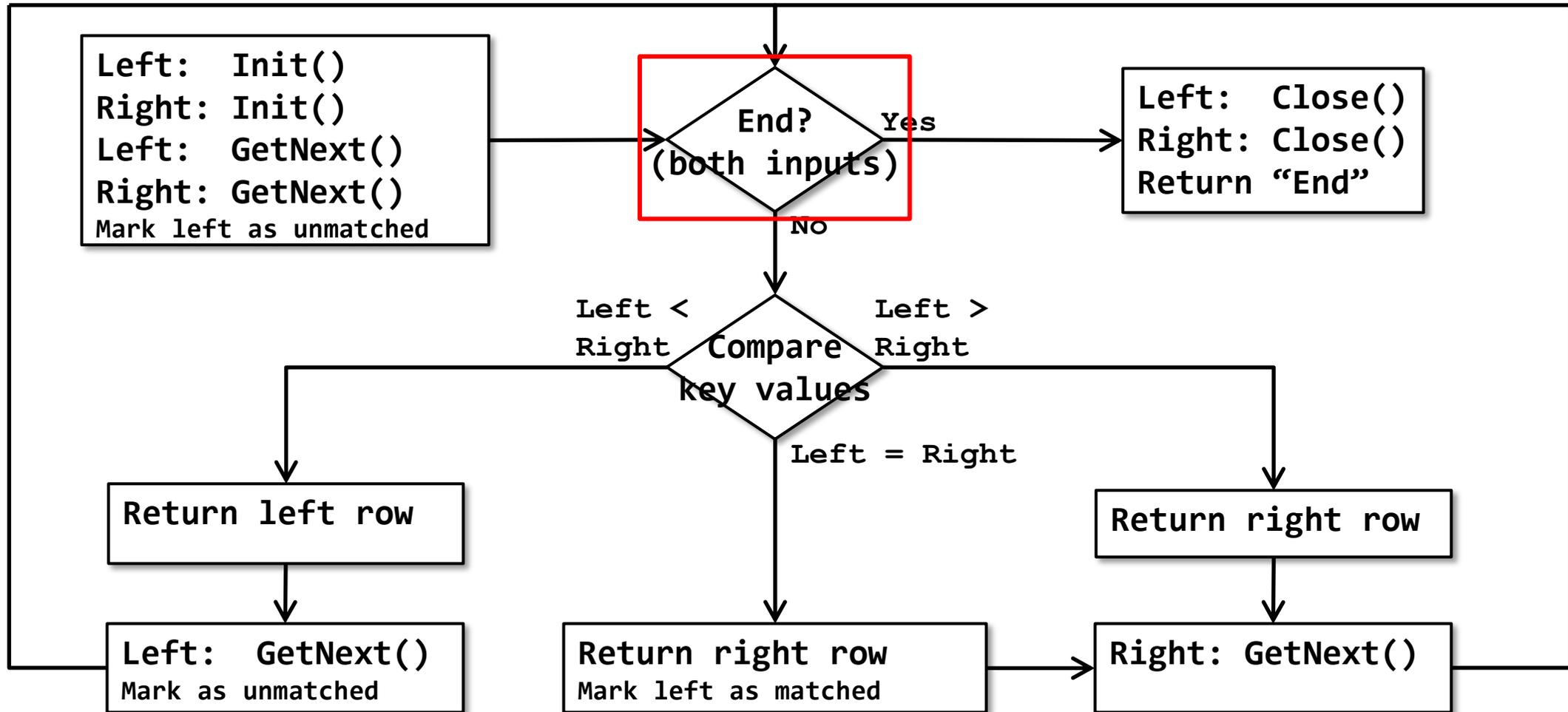
# Merge Join (inner join to concatenation)



# Merge Join (inner join to concatenation)



# Merge Join (concatenation, one to many)



# Merge Join

## Merge Join

Logical operations supported

Inner Join

Left / Right / Full Outer Join

Left / Right Semi Join

Left / Right Anti Semi Join

Probed Left Semi Join

Concatenation → Similar to UNION ALL in T-SQL

Union → Similar to UNION in T-SQL



# Merge Join (concatenation to union)

Left

KeyCol
1
1
3

Right

KeyCol
2
2
3

Concatenation

KeyCol
1
1
2
2
3
3

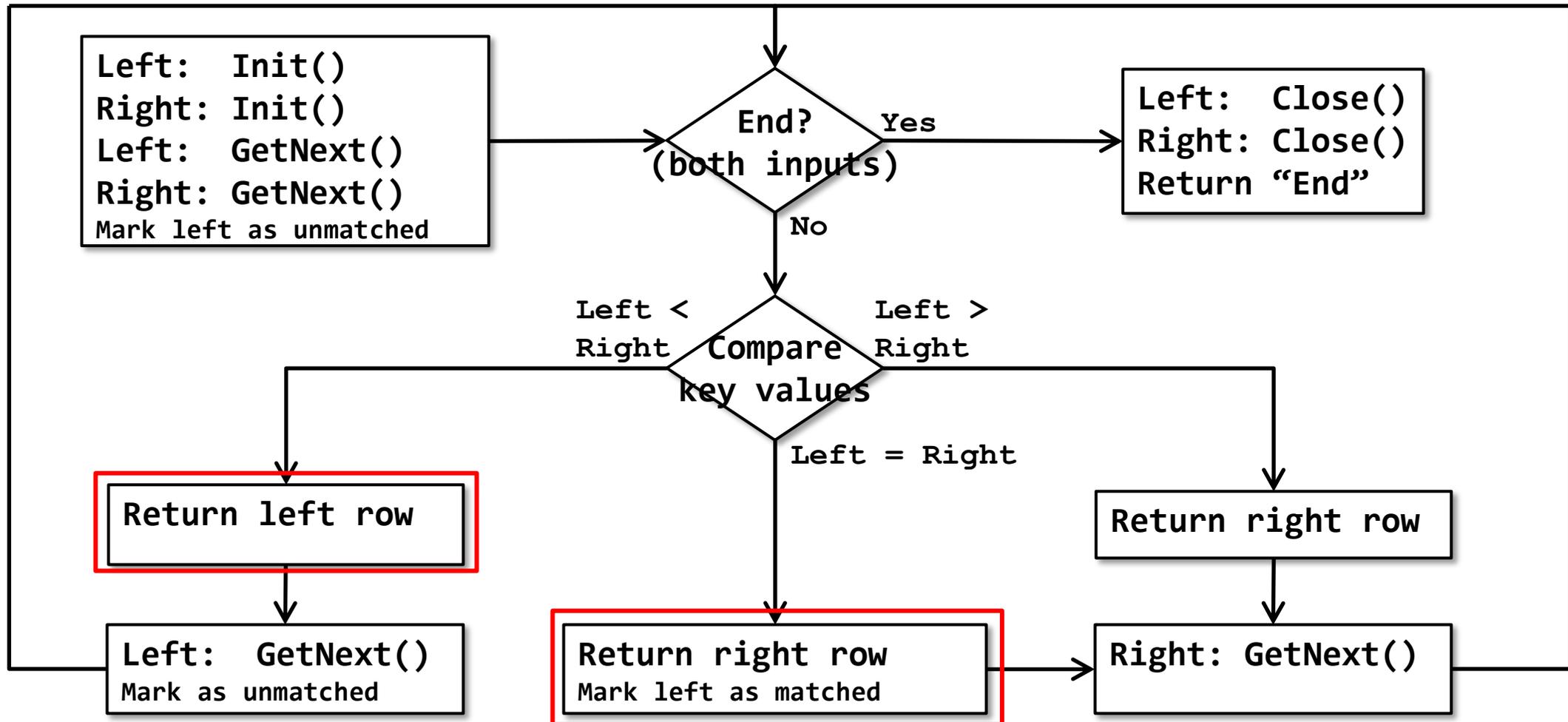
Union

KeyCol
1
2
3

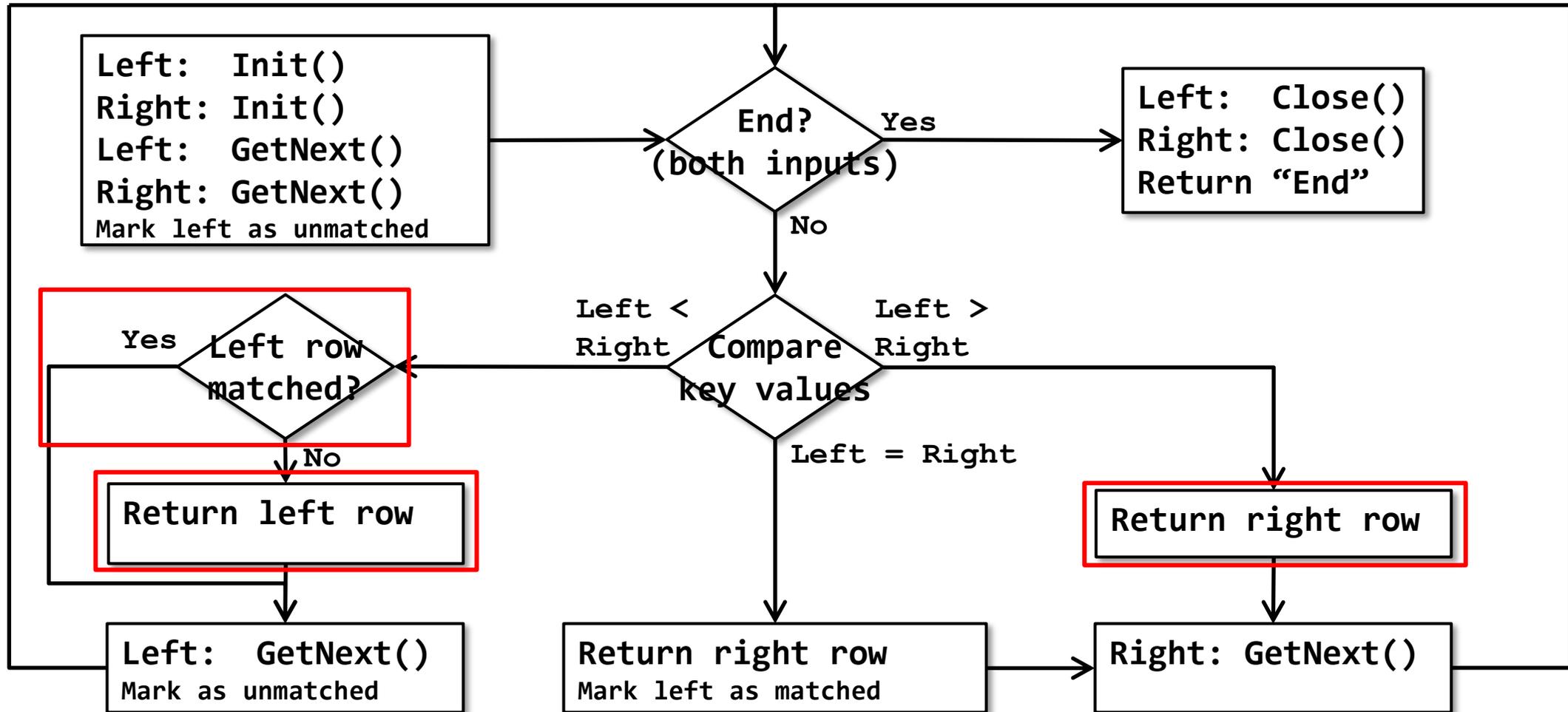


Merge Join  
(Inner Join)

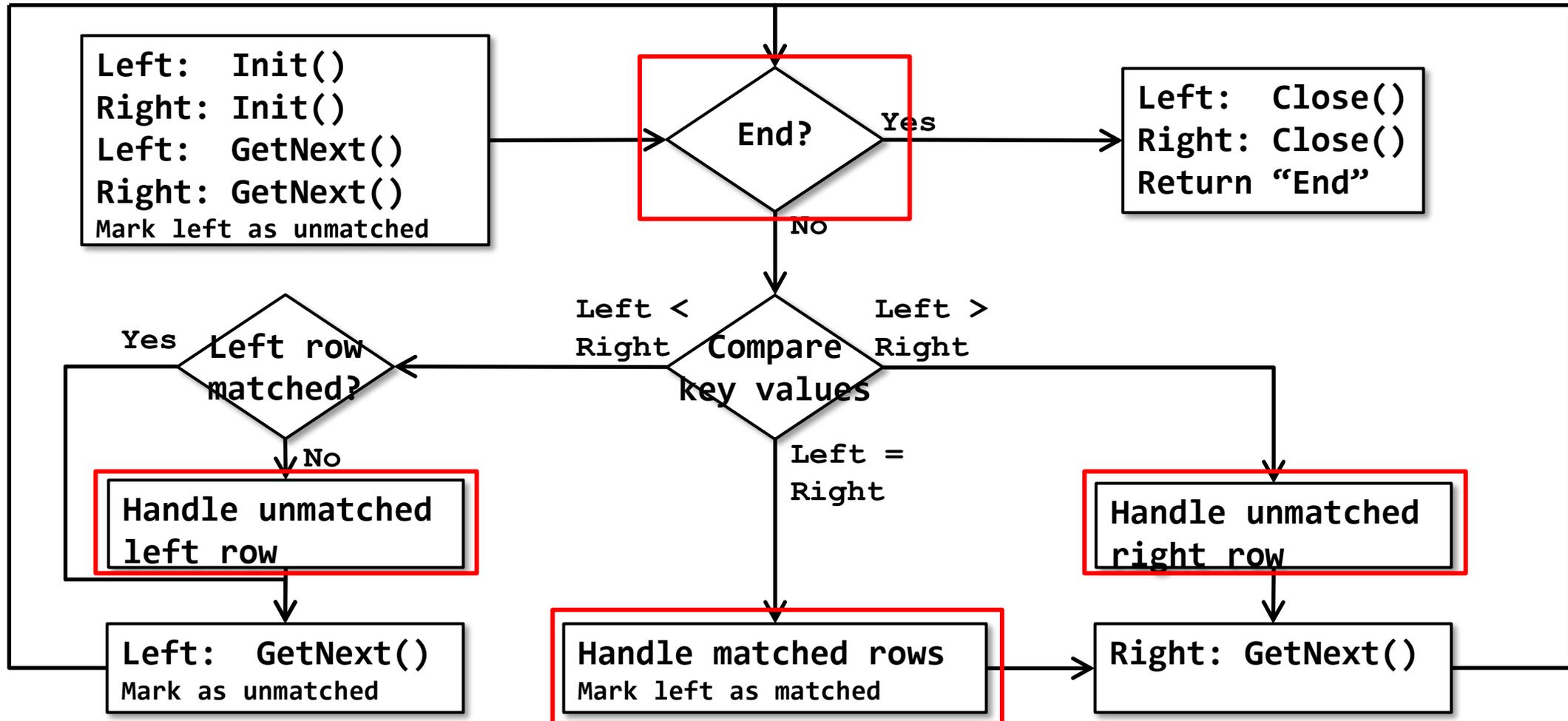
# Merge Join (concatenation to union)



# Merge Join (union, one to many)



# Merge Join (all operations, one to many)



# Merge Join

*Many to Many = True*

→

FruitNum	FruitName
1	Apple
2	Lime
2	Orange
3	Cherry
3	Melon

→

ColorNum	ColorName	ColorCode
0	Black	#FFFFFF
1	Red	#FF0000
2	Orange	#FFA500
2	Yellow	#FFFF00
4	Blue	#0000FF
5	White	#FFFFFF

FruitNum	FruitName	ColorName
1	Apple	Red
2	Lime	Orange
2	Lime	Yellow

# Merge Join

*Many to Many = True*

→

FruitNum	FruitName
1	Apple
2	Lime
2	Orange
3	Cherry
3	Melon

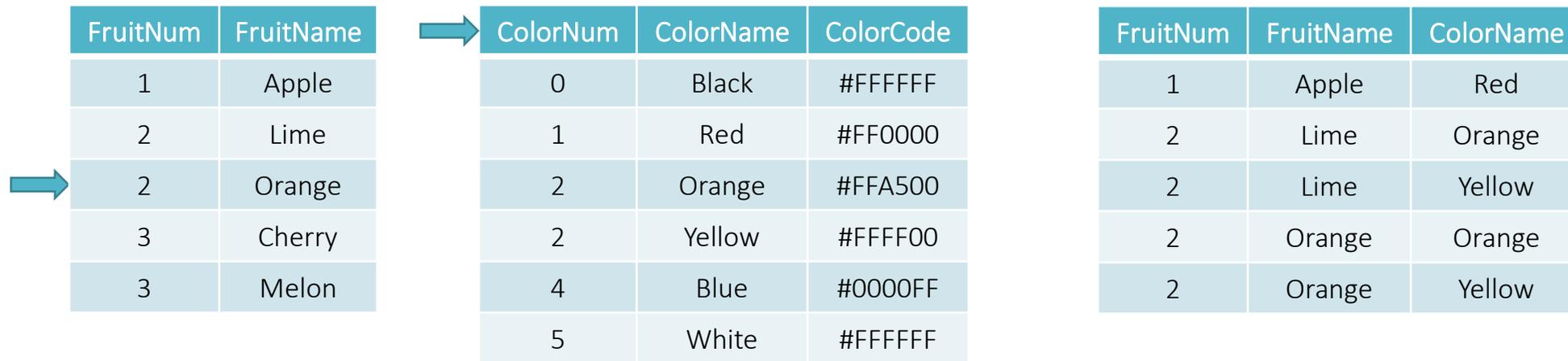
→

ColorNum	ColorName	ColorCode
0	Black	#FFFFFF
1	Red	#FF0000
2	Orange	#FFA500
2	Yellow	#FFFF00
4	Blue	#0000FF
5	White	#FFFFFF

FruitNum	FruitName	ColorName
1	Apple	Red
2	Lime	Orange
2	Lime	Yellow

# Merge Join

*Many to Many = True*



# Merge Join

*Many to Many = True*

Worktable

ColorNum	ColorName
2	Orange
2	Yellow

FruitNum	FruitName
1	Apple
2	Lime
2	Orange
3	Cherry
3	Melon

ColorNum	ColorName	ColorCode
0	Black	#FFFFFF
1	Red	#FF0000
2	Orange	#FFA500
2	Yellow	#FFFF00
4	Blue	#0000FF
5	White	#FFFFFF

FruitNum	FruitName	ColorName
1	Apple	Red
2	Lime	Orange
2	Lime	Yellow
2	Orange	Orange
2	Orange	Yellow

# Merge Join

*Many to Many = True*  
May swap inputs

Worktable

ColorNum	ColorName
2	Orange
2	Yellow

FruitNum	FruitName
1	Apple
2	Lime
2	Orange
3	Cherry
3	Melon

ColorNum	ColorName	ColorCode
0	Black	#FFFFFF
1	Red	#FF0000
2	Orange	#FFA500
2	Yellow	#FFFF00
4	Blue	#0000FF
5	White	#FFFFFF

FruitNum	FruitName	ColorName
1	Apple	Red
2	Lime	Orange
2	Lime	Yellow
2	Orange	Orange
2	Orange	Yellow

# Merge Join

*Many to Many* = True

May swap inputs

Cost of tempdb *writes*:

- Estimated number of rows in right input

- Estimated size of rows in right input

- Estimated percentage of duplicates in right input

Cost of tempdb *reads*:

- Estimated number of left rows that match a non-unique right row

- Estimated number and size of matching right rows for those left rows

# Merge Join

*Many to Many* = True

May swap inputs

Effect on order:

For join columns, still order-retaining

With the usual exceptions for Left Outer Join, Right Outer Join, and Full Outer Join

More complex when the known sort order spans more columns

Column 1	Column 2
1	C
2	B
3	A

(Column 1, Column 2)

Column 1	Column 3
1	25
2	8
2	13

(Column 1, Column 3)

Column 1	Column 2	Column 3
1	C	25
2	B	8
2	B	13

(Column 1, Column 2)  
(Column 1, Column 3)

# Merge Join

*Many to Many* = True

May swap inputs

Effect on order:

For join columns, still order-retaining

With the usual exceptions for Left Outer Join, Right Outer Join, and Full Outer Join

More complex when the known sort order spans more columns

Sort order of left input retained, sort order of right input is lost

Column 1	Column 2
1	C
2	A
2	B

(Column 1, Column 2)

Column 1	Column 3
1	25
2	8
2	13

(Column 1, Column 3)

Column 1	Column 2	Column 3
1	C	25
2	A	8
2	A	13
2	B	8
2	B	13

(Column 1, Column 2)

~~(Column 1, Column 3)~~

# Merge Join

*Many to Many* = True

May swap inputs

Effect on order

Requires tracking of matched / unmatched for row in worktable

Probably stored with the data in the worktable

Results in extra writes to worktable to update this Boolean

Requires extra read through worktable after last matching left row

# Merge Join

*Many to Many* = True

May swap inputs

Effect on order

Requires tracking of matched / unmatched for row in worktable

Only relevant for “true” join types

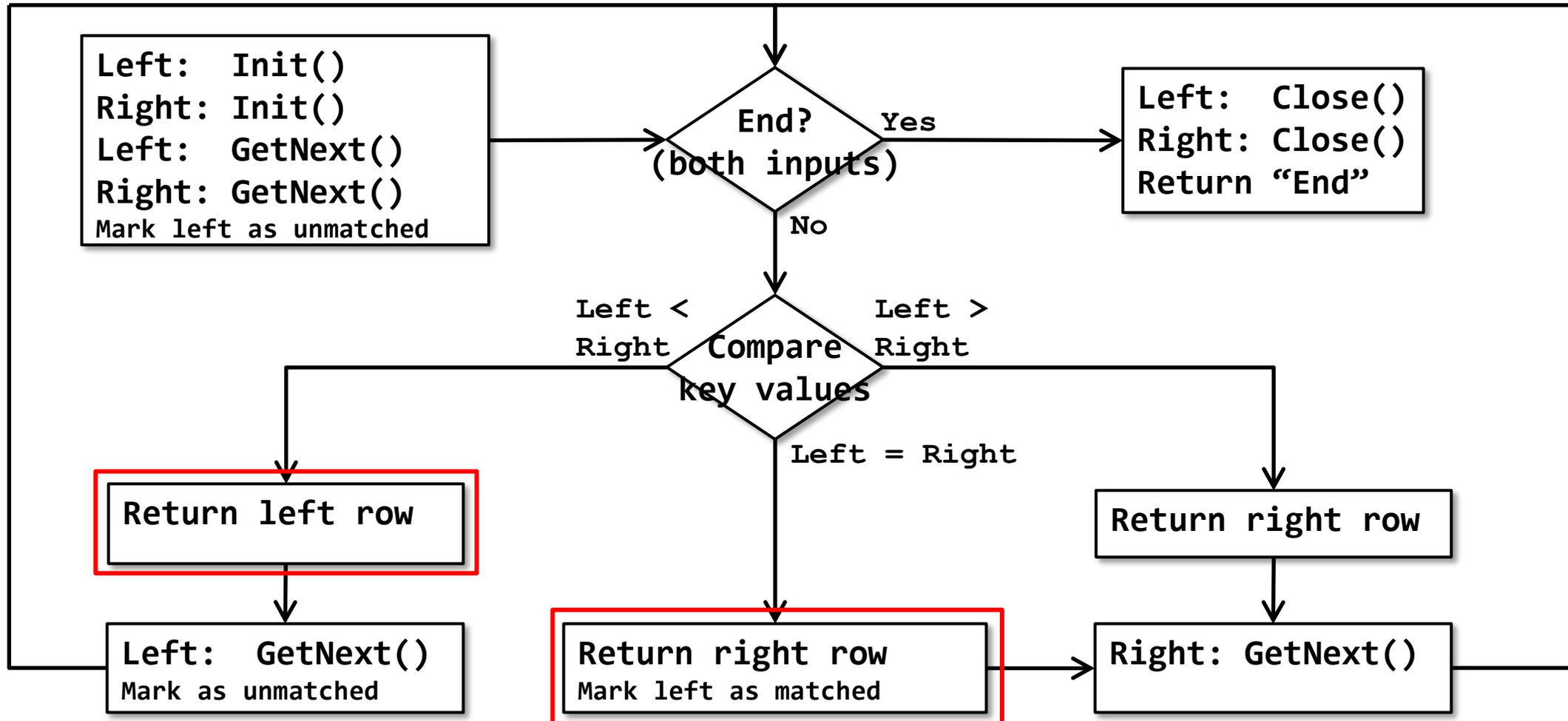
Inner Join, Left / Right / Full Outer Join, (probed) Left / Right (Anti) Semi Join

Not relevant for Union and Concatenation

Union: Requires uniqueness in both inputs (one to one)

Concatenation: Duplicates in either inputs don't affect results

# Merge Join (concatenation, one to many)



# Merge Join

*Many to Many* = True

May swap inputs

Effect on order

Requires tracking of matched / unmatched for row in worktable

Only relevant for “true” join types

Inner Join, Left / Right / Full Outer Join, (probed) Left / Right (Anti) Semi Join

Not relevant for Union and Concatenation

No *Many to Many* property for Union and Concatenation

# Merge Join

*Many to Many* = True

May swap inputs

Effect on order

Requires tracking of matched / unmatched for row in worktable

Only relevant for “true” join types

May (in rare cases) be used for Cartesian product

*Where (join columns)* property:  $() = ()$

(Empty set is equal to empty set)

Effectively, this is always true, so each row matches each other row

Often combined with inequality predicate in *Residual* property

# Summary

## Merge Join (advanced)

Logic for all supported join types

Worktable for many to many

Edge case for Cartesian product

# Next chapters

## Chapter 3: Hash Match (advanced)

- Logic for all supported join types

- Memory usage

- Spill internals

- BitmapCreator* property

## Chapter 4: Adaptive Join