Exercise 39: Walking trees

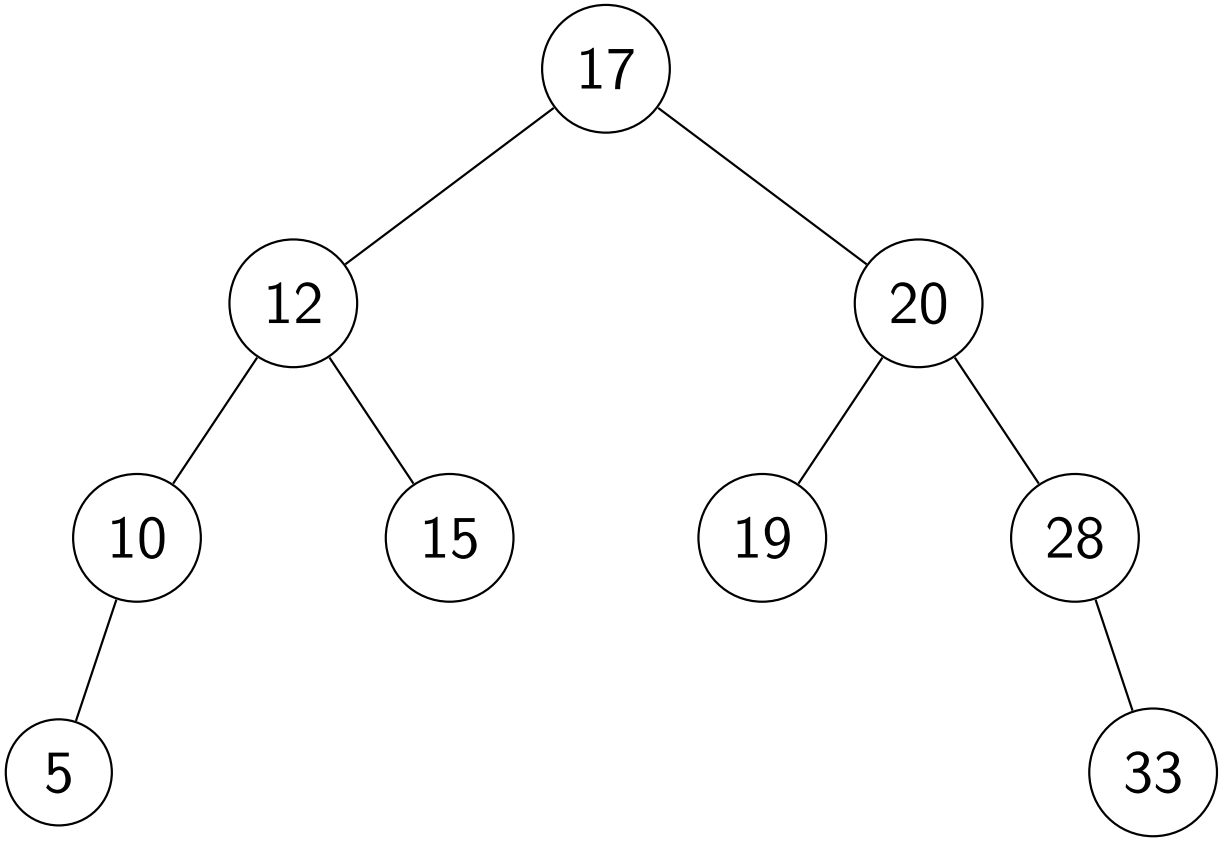
Andreas Loibl

December 1, 2010

Traversal of trees

Pre-Order (output - left - right)

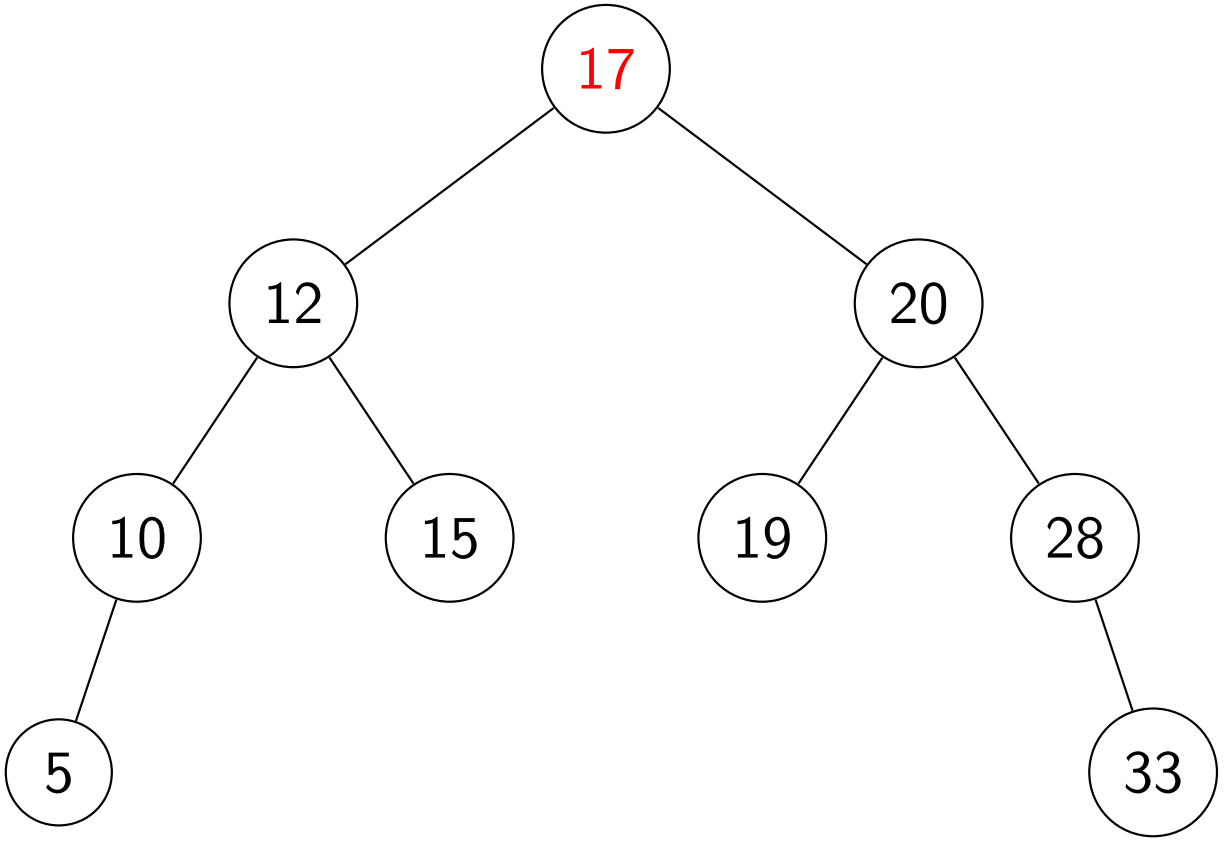
Tree:



Output:

Pre-Order (output - left - right)

Tree:

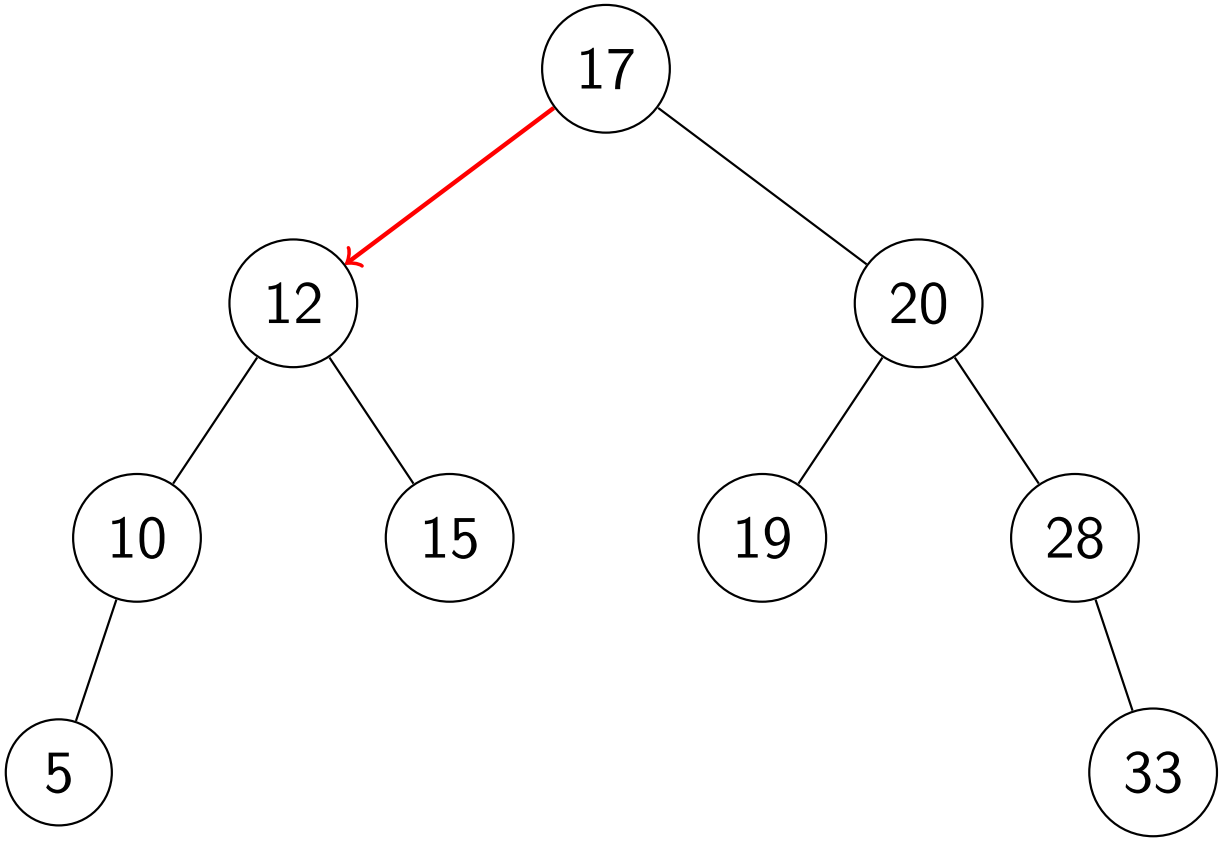


Output:

17

Pre-Order (output - left - right)

Tree:

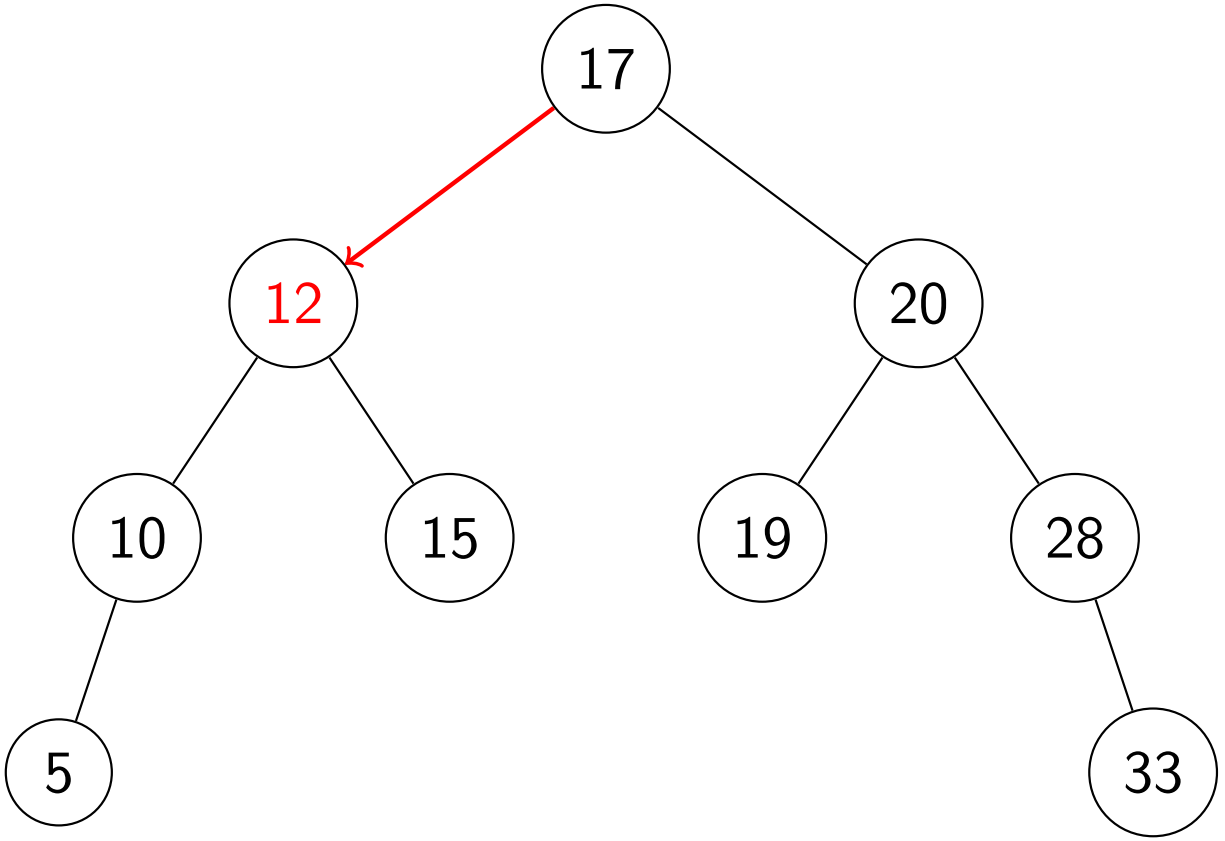


Output:

17

Pre-Order (output - left - right)

Tree:

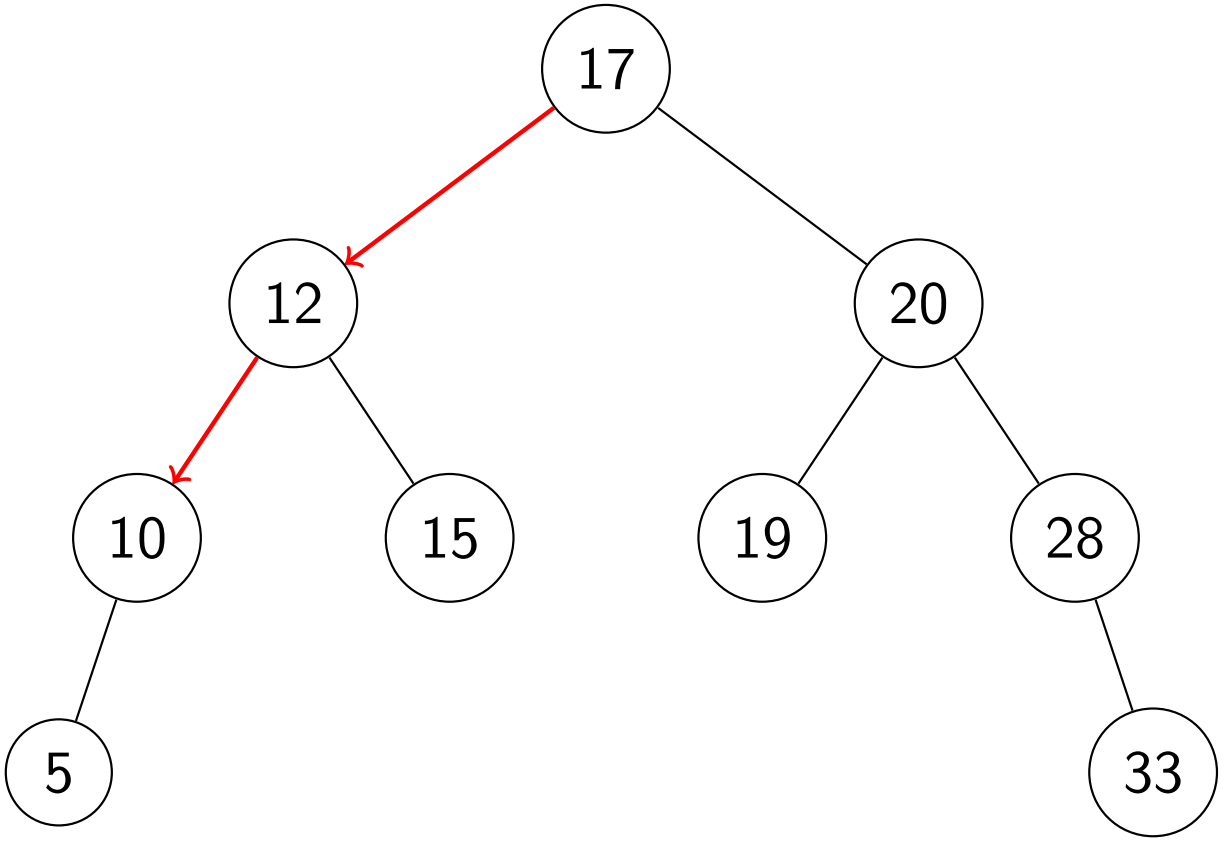


Output:

17
12

Pre-Order (output - left - right)

Tree:

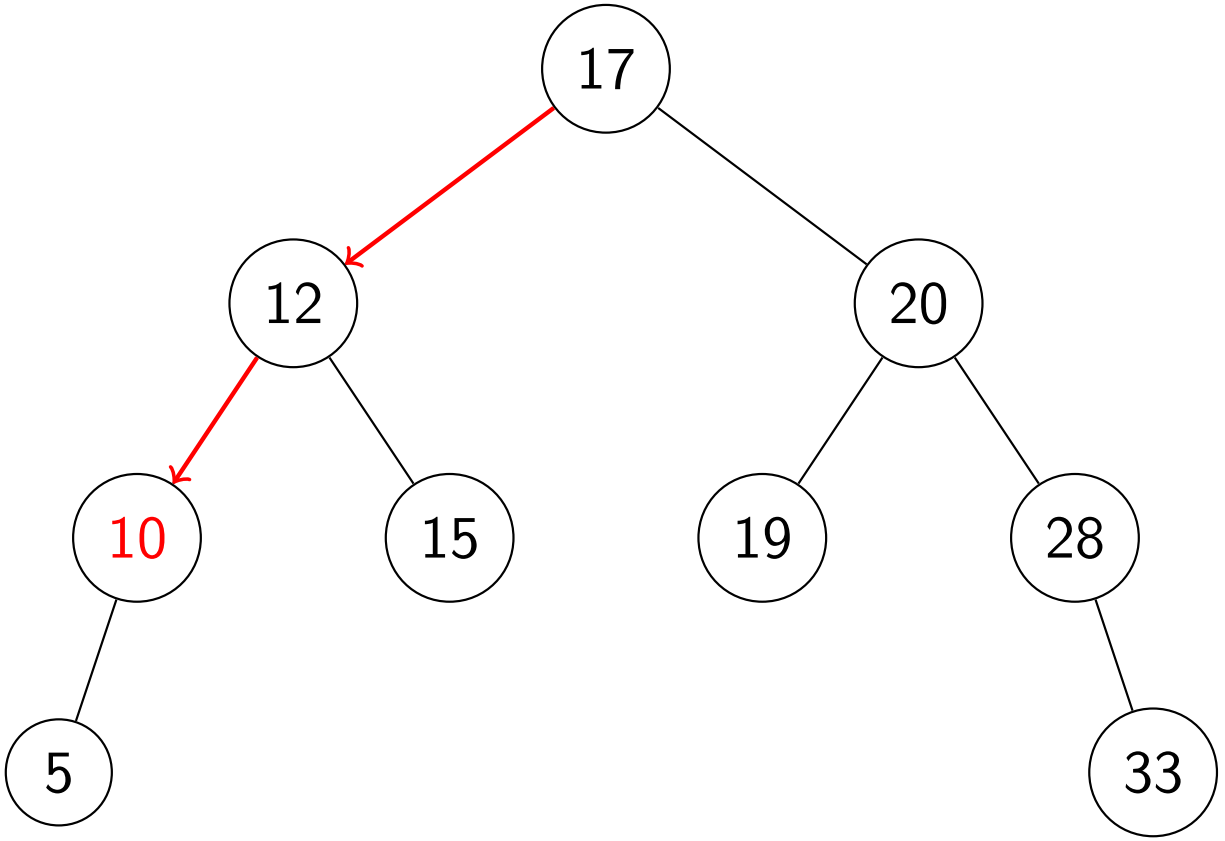


Output:

17
12

Pre-Order (output - left - right)

Tree:

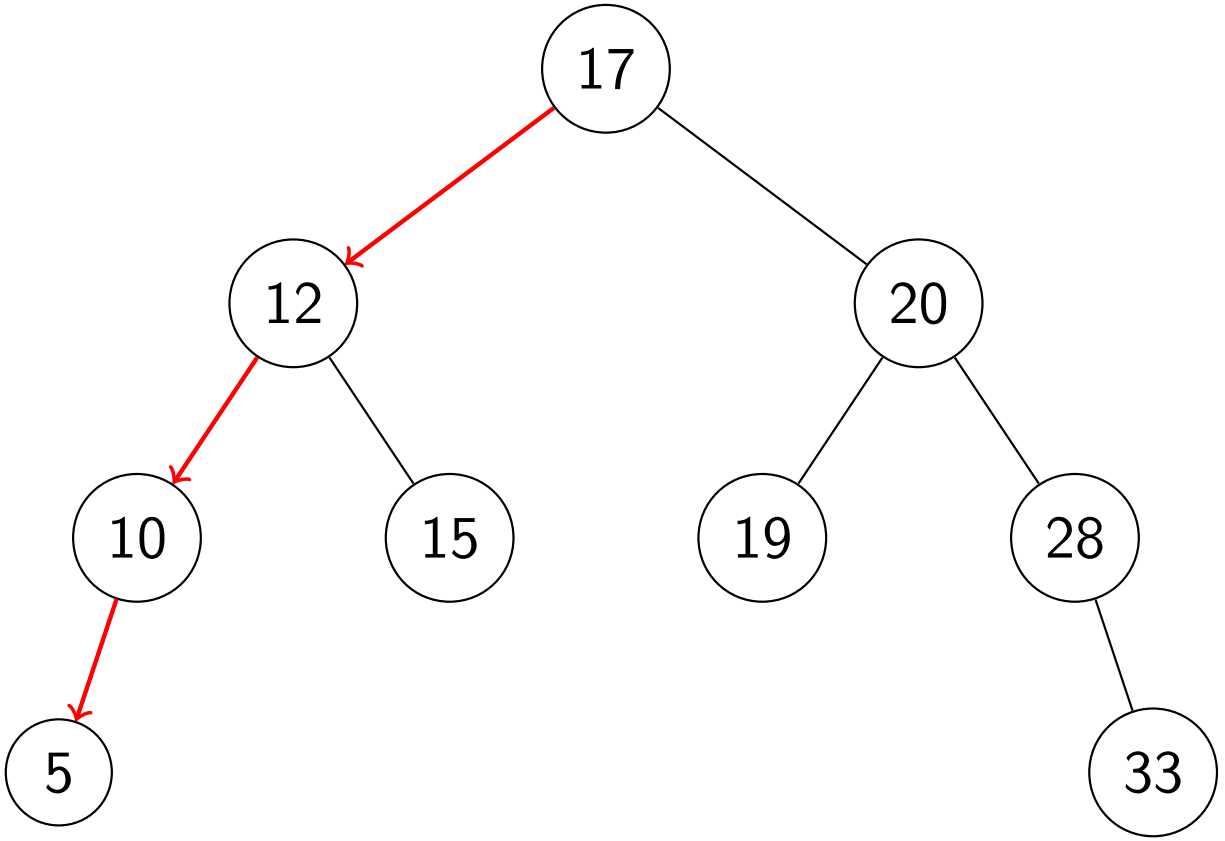


Output:

17
12
10

Pre-Order (output - left - right)

Tree:

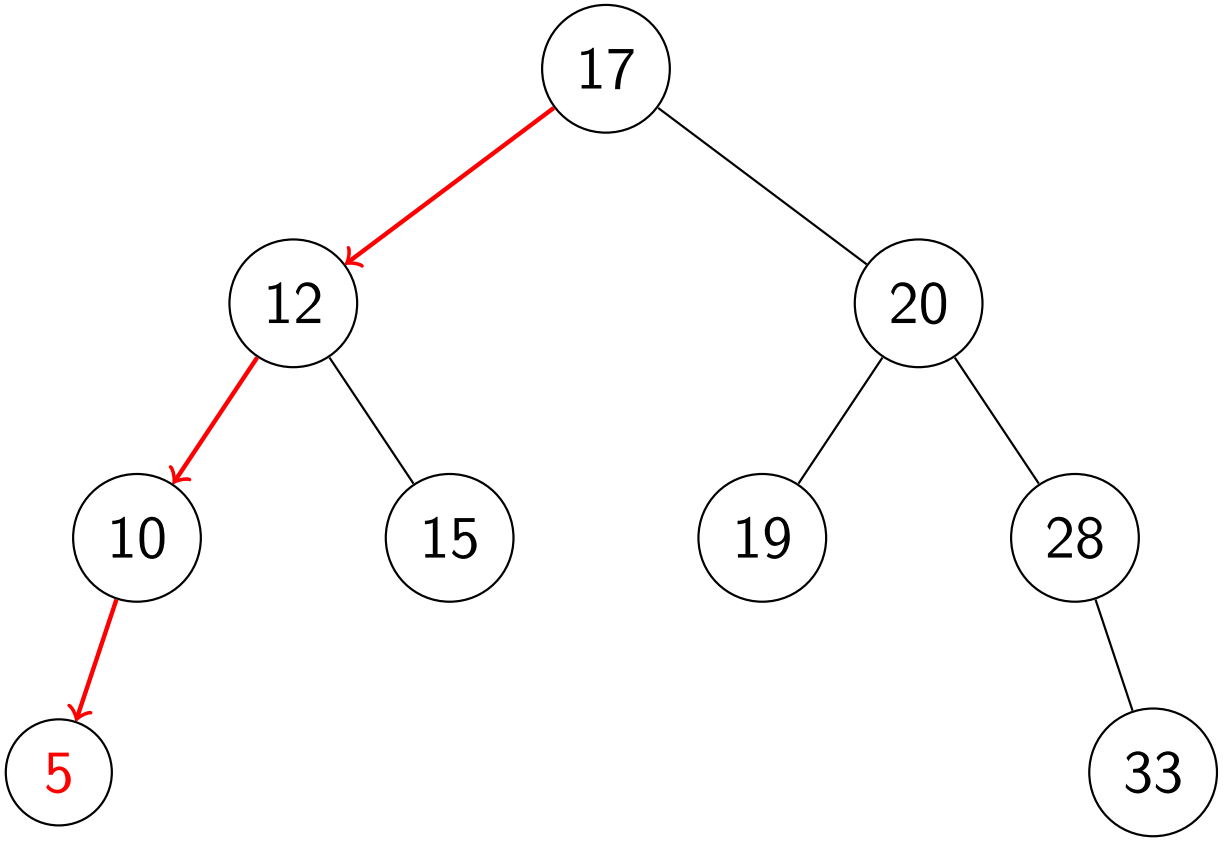


Output:

17
12
10

Pre-Order (output - left - right)

Tree:

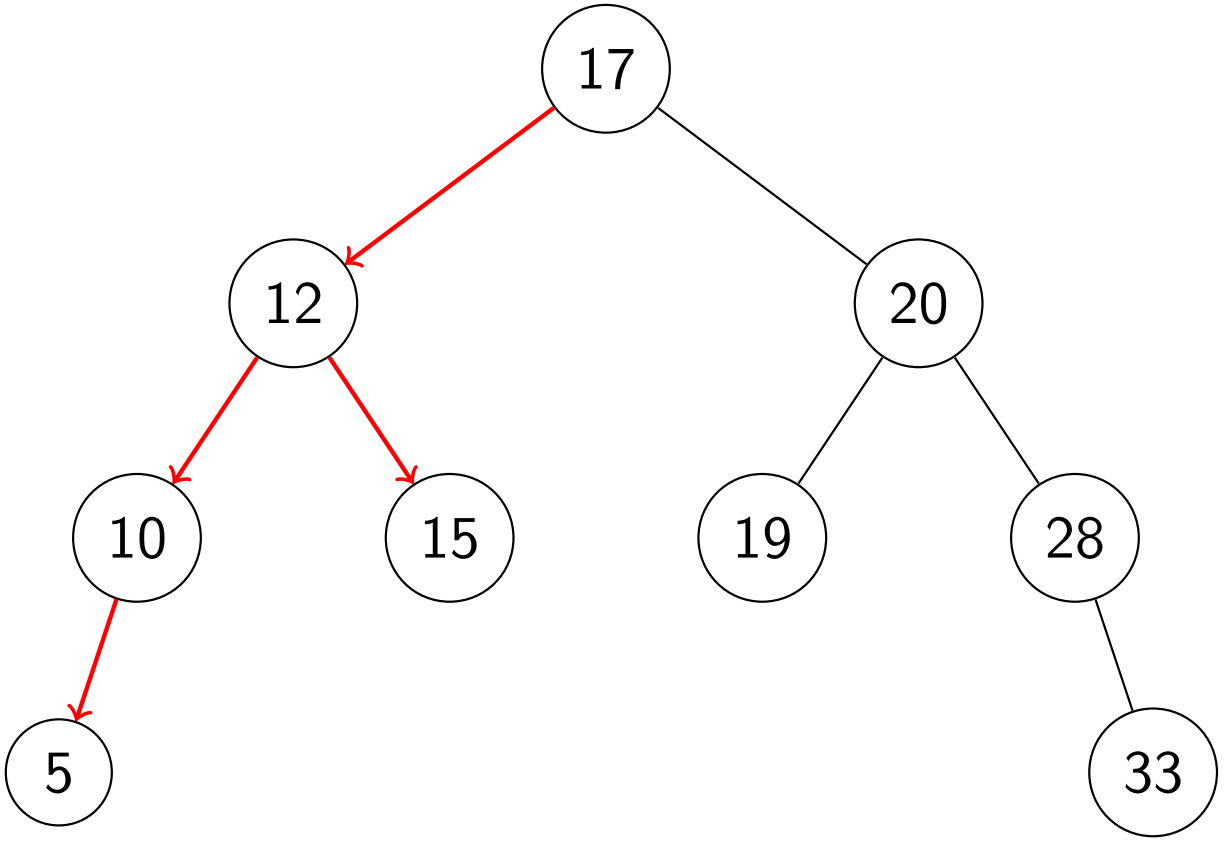


Output:

17
12
10
5

Pre-Order (output - left - right)

Tree:

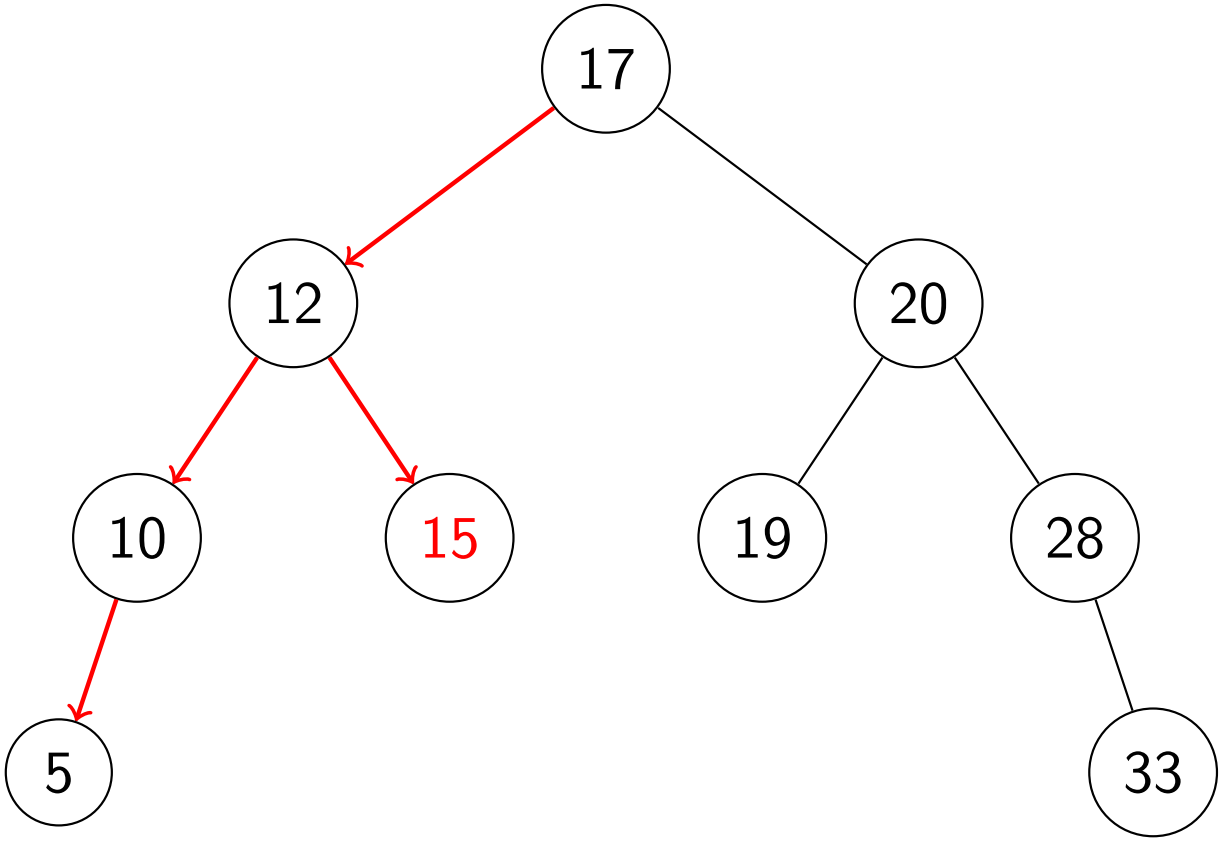


Output:

17
12
10
5

Pre-Order (output - left - right)

Tree:

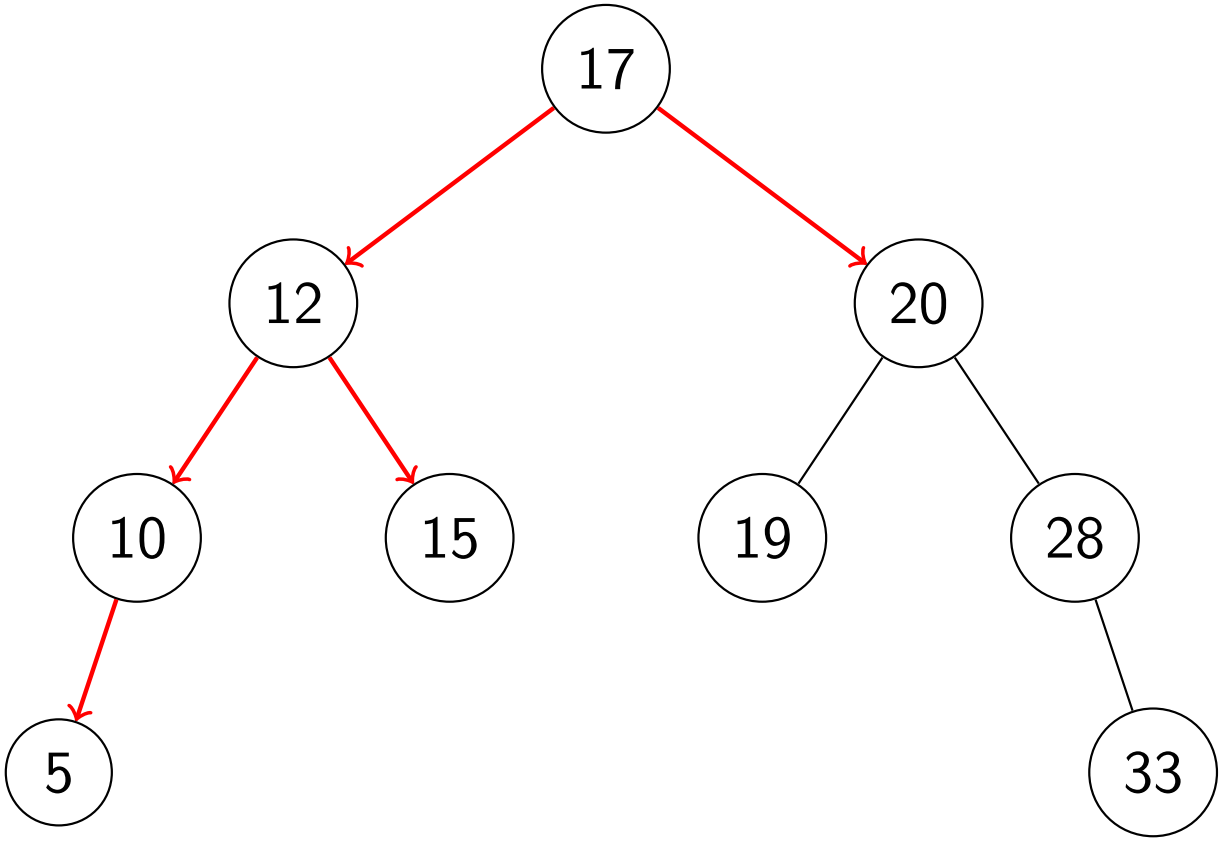


Output:

17
12
10
5
15

Pre-Order (output - left - right)

Tree:

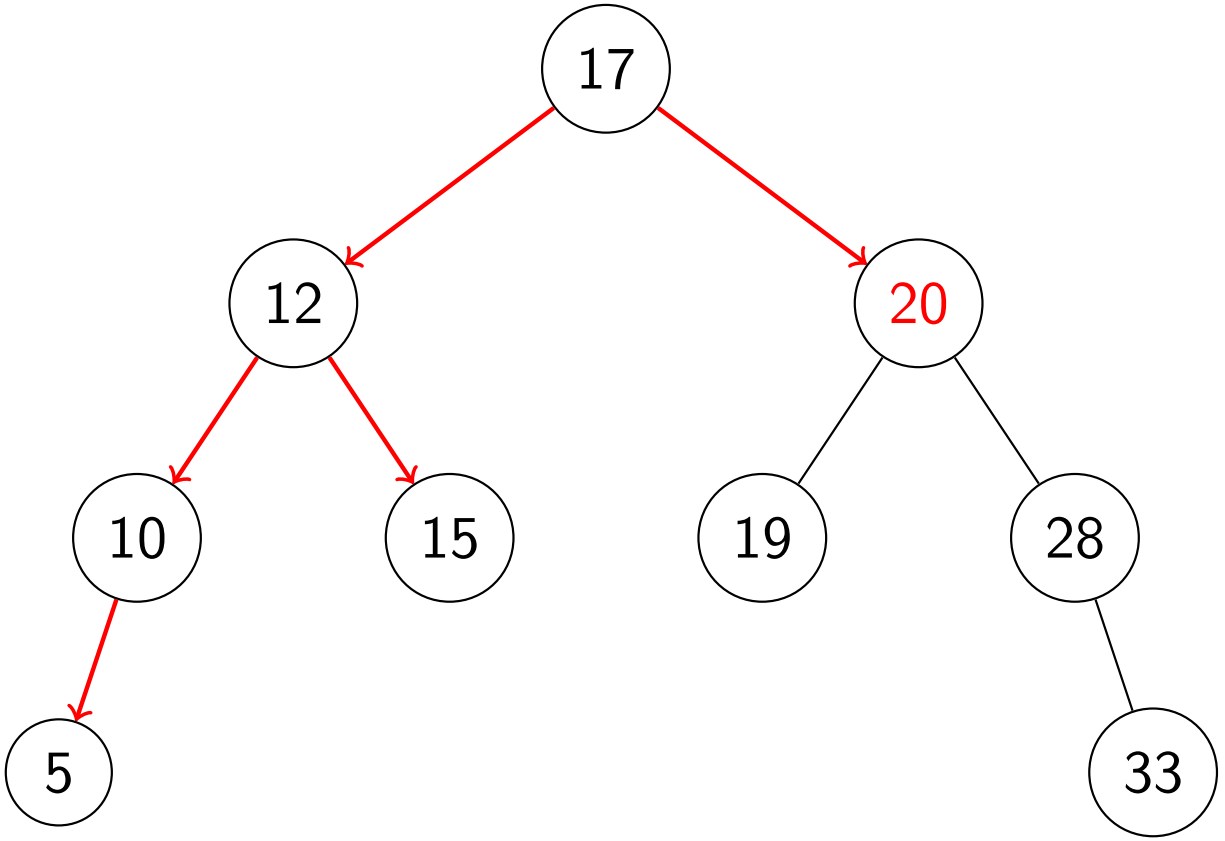


Output:

17
12
10
5
15

Pre-Order (output - left - right)

Tree:

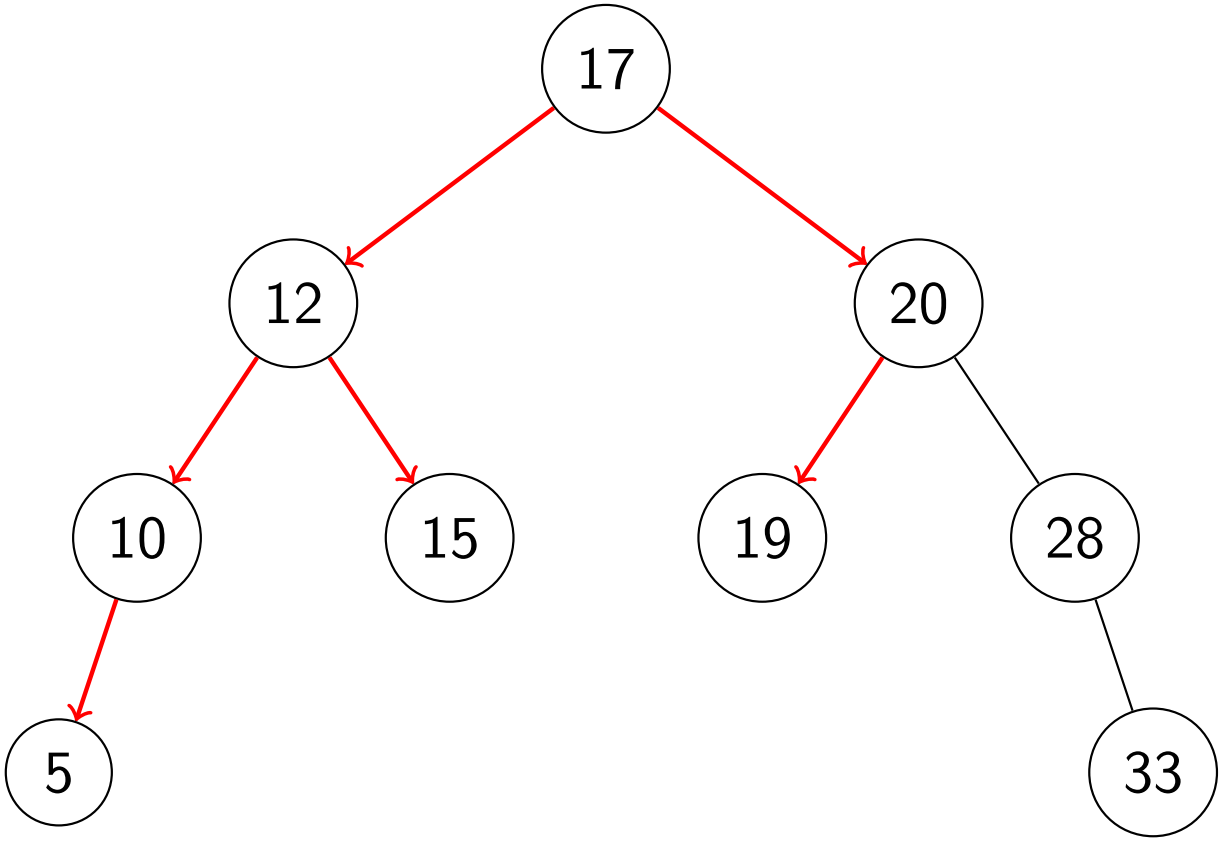


Output:

17
12
10
5
15
20

Pre-Order (output - left - right)

Tree:

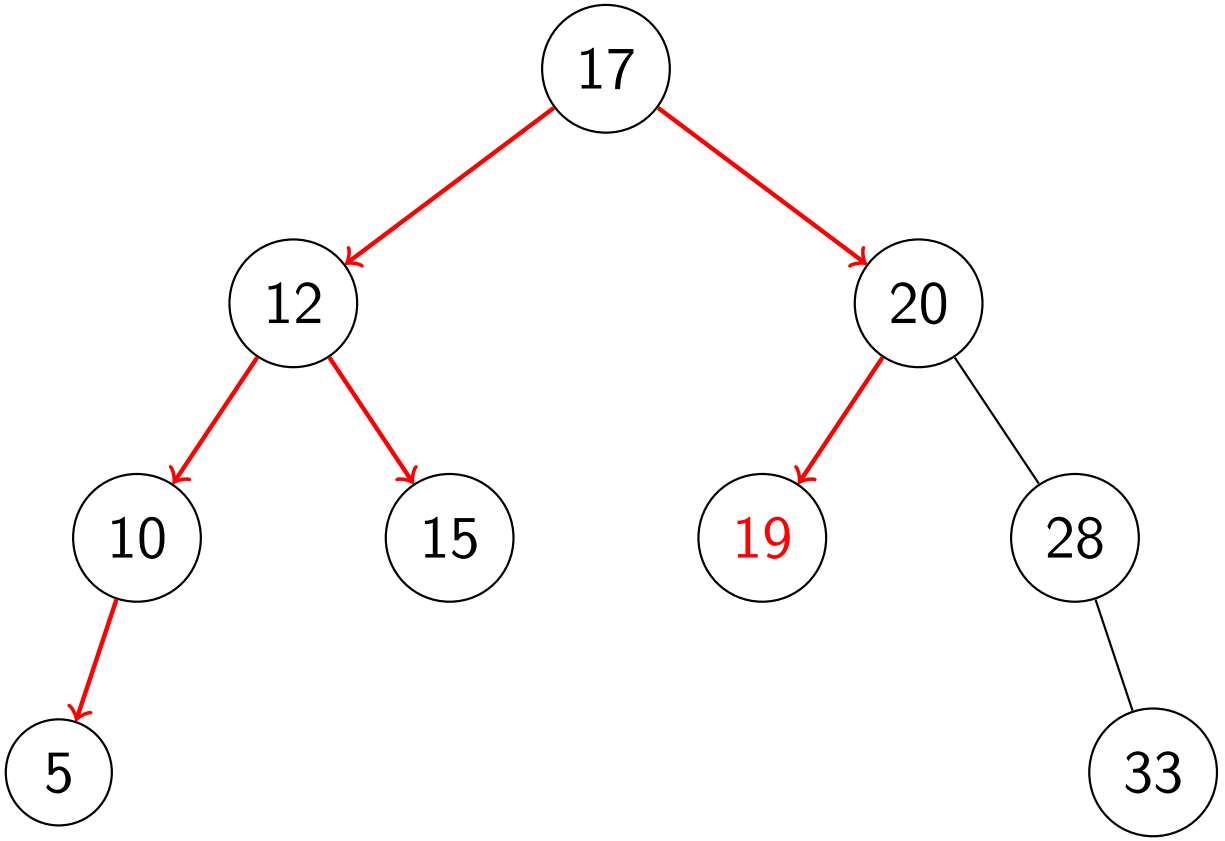


Output:

17
12
10
5
15
20

Pre-Order (output - left - right)

Tree:

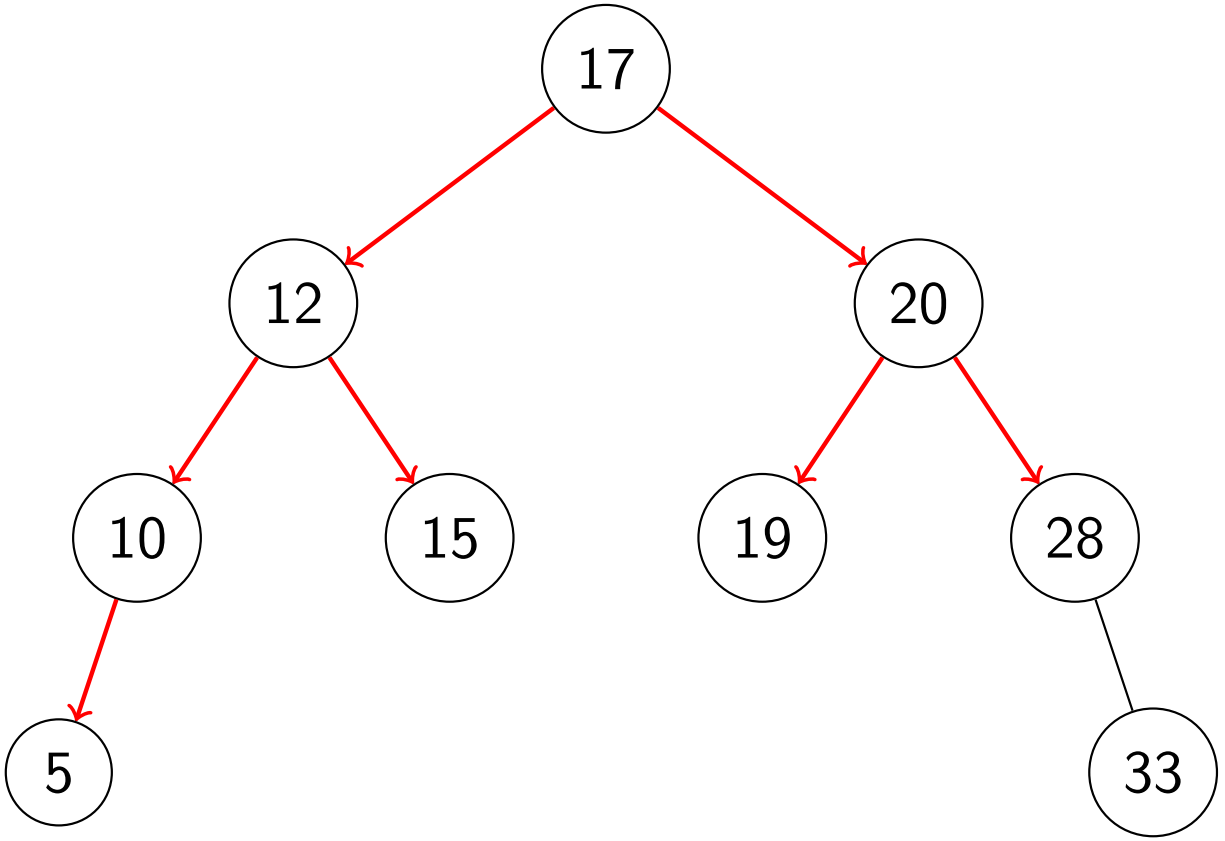


Output:

17
12
10
5
15
20
19

Pre-Order (output - left - right)

Tree:

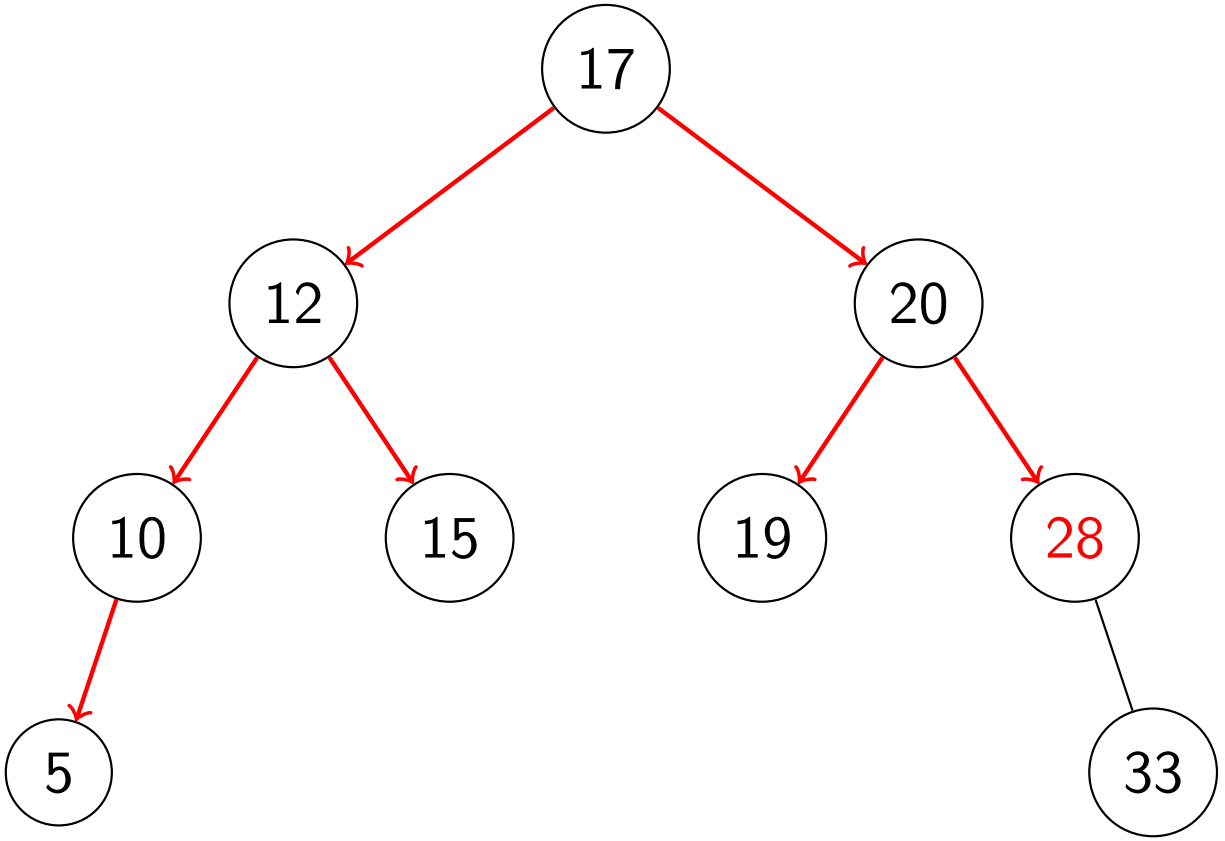


Output:

17
12
10
5
15
20
19

Pre-Order (output - left - right)

Tree:

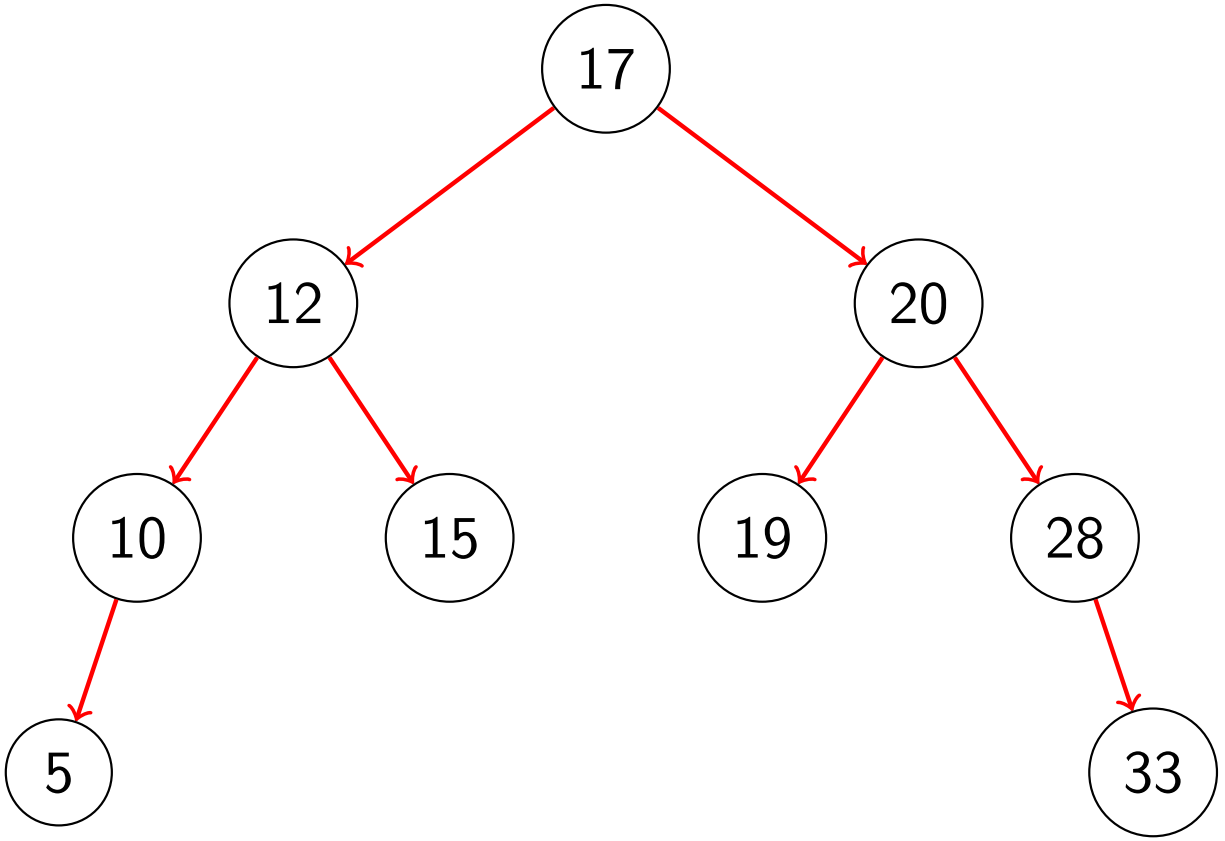


Output:

17
12
10
5
15
20
19
28

Pre-Order (output - left - right)

Tree:

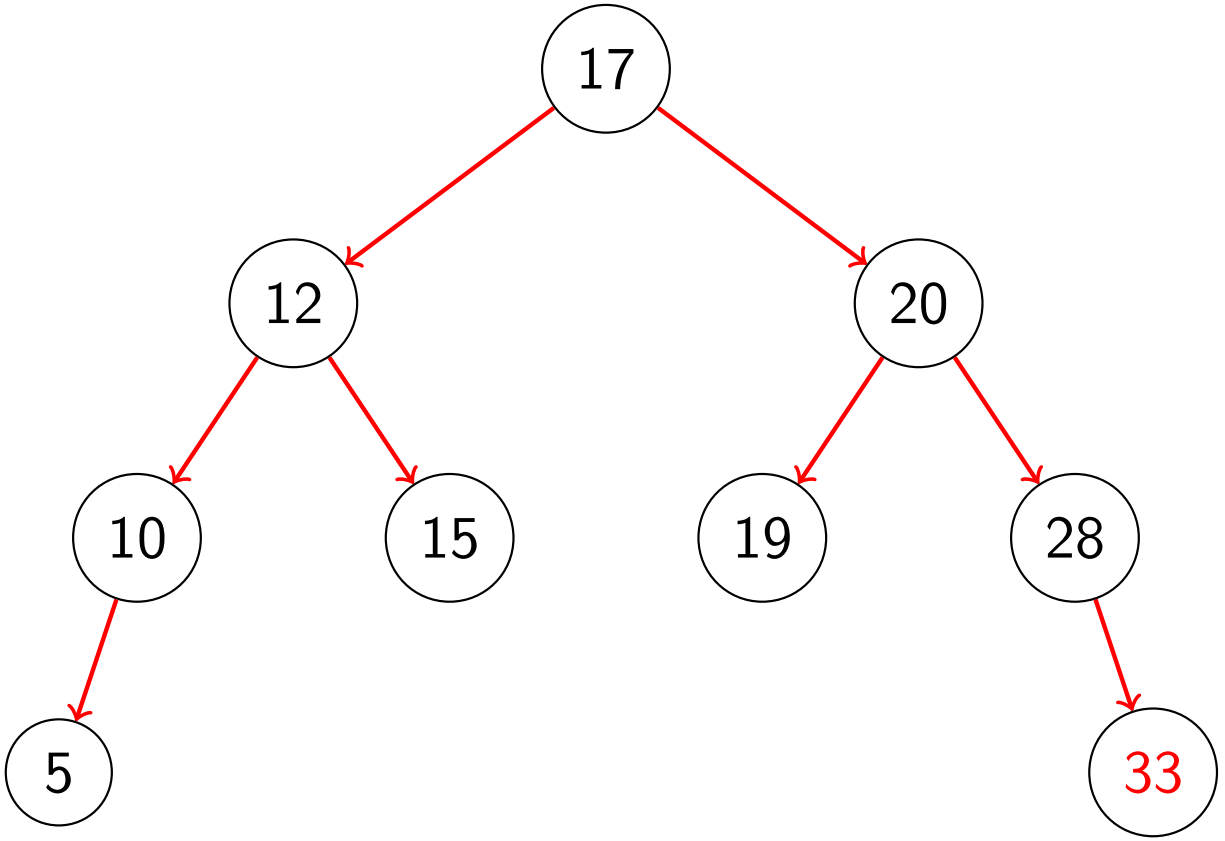


Output:

17
12
10
5
15
20
19
28

Pre-Order (output - left - right)

Tree:

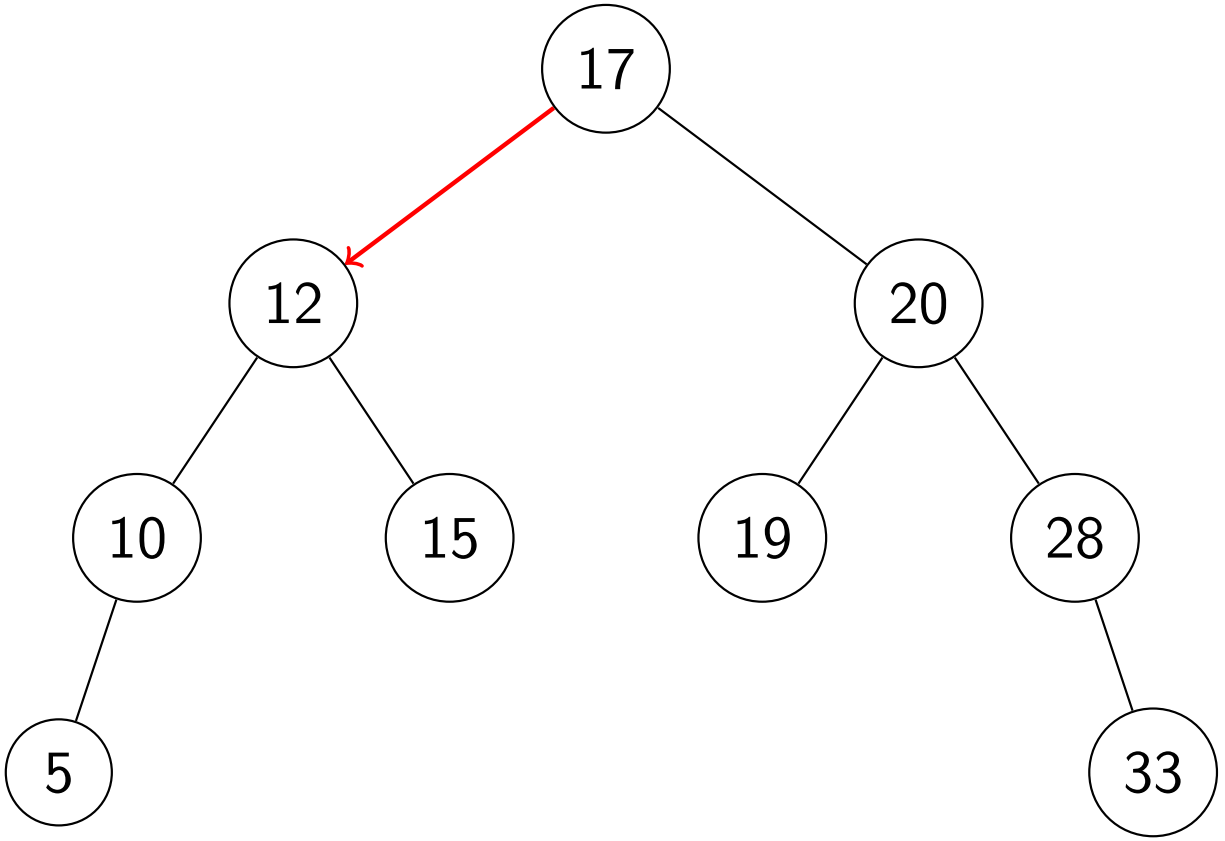


Output:

17
12
10
5
15
20
19
28
33

In-Order (left - output - right)

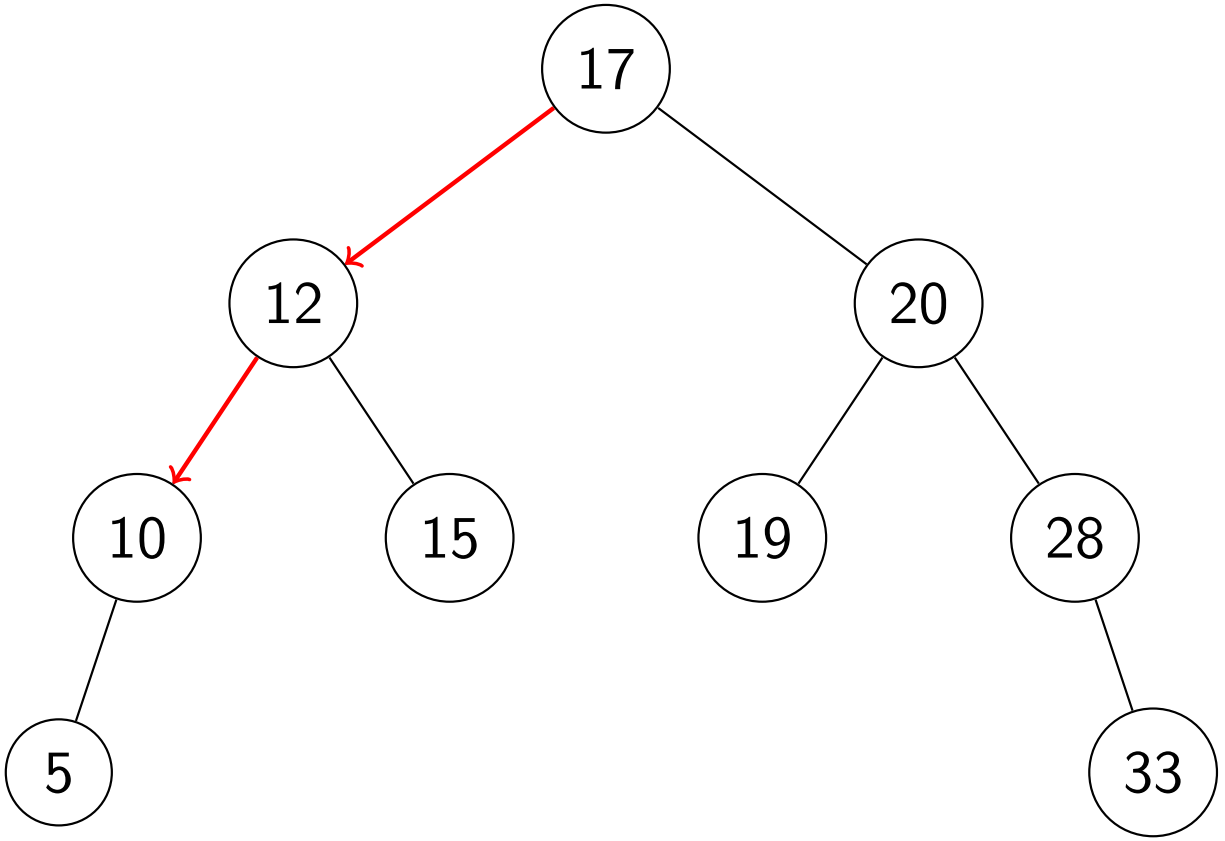
Tree:



Output:

In-Order (left - output - right)

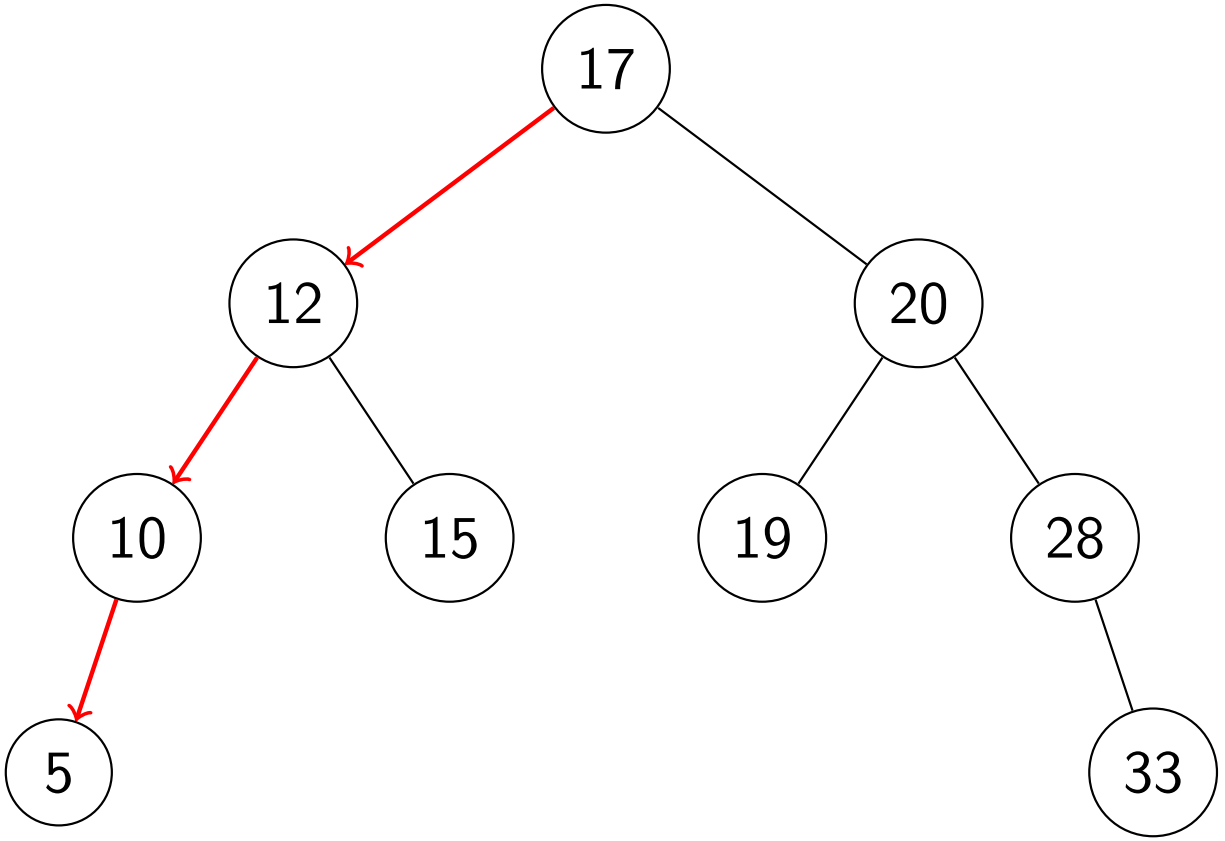
Tree:



Output:

In-Order (left - output - right)

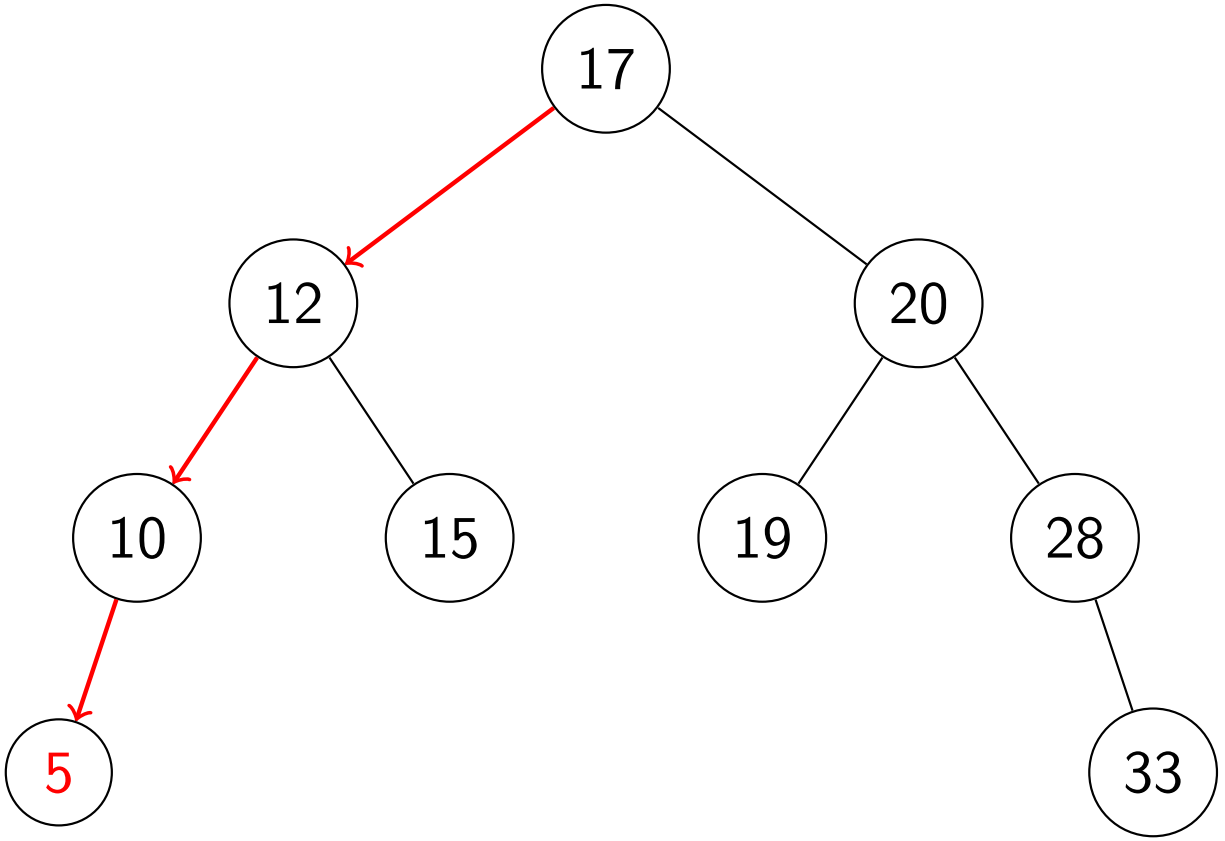
Tree:



Output:

In-Order (left - output - right)

Tree:

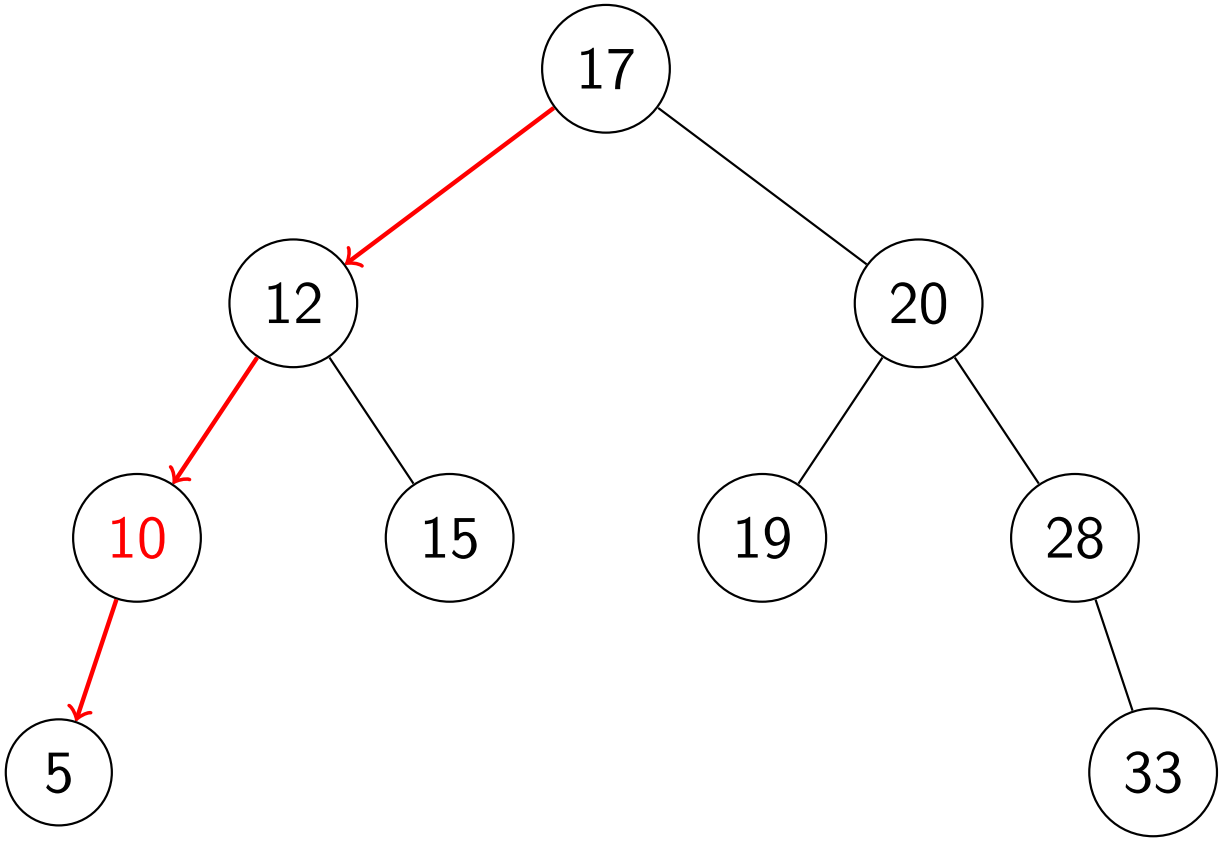


Output:

5

In-Order (left - output - right)

Tree:

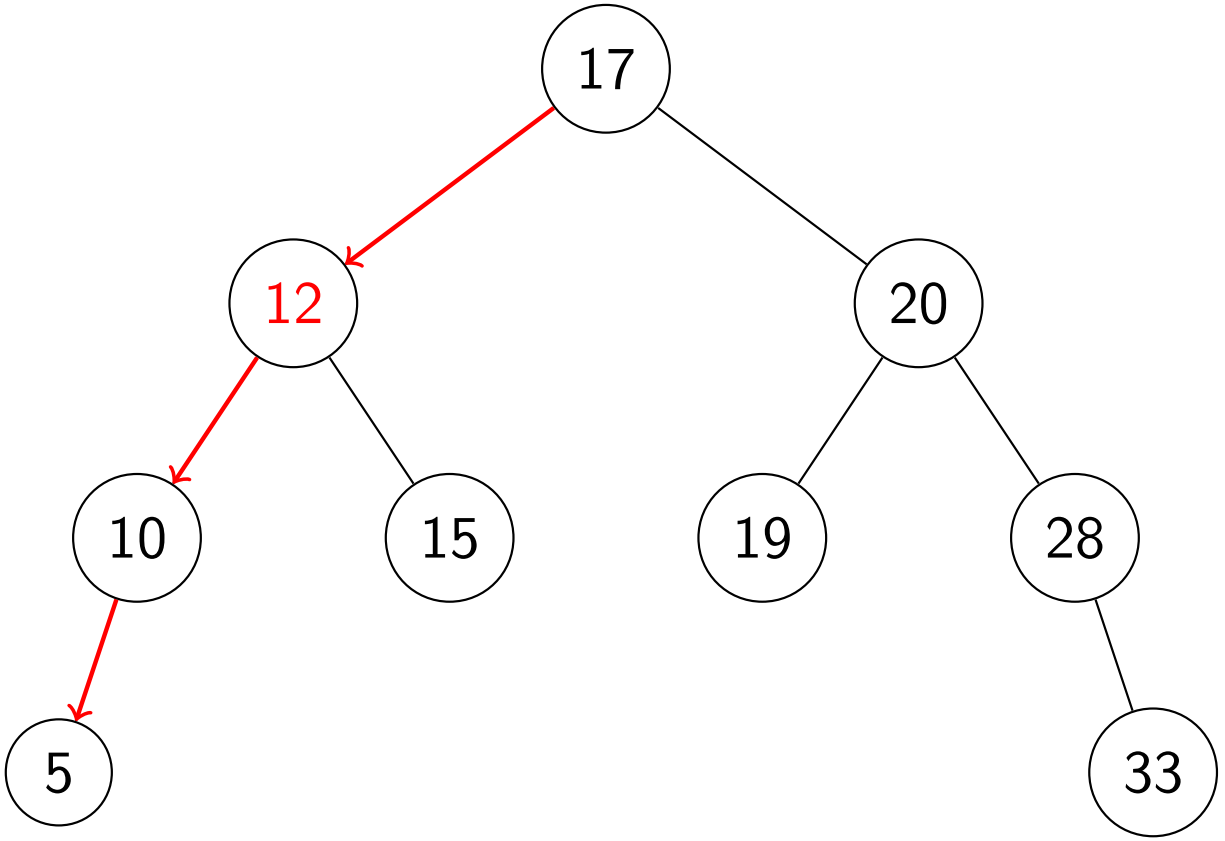


Output:

5
10

In-Order (left - output - right)

Tree:

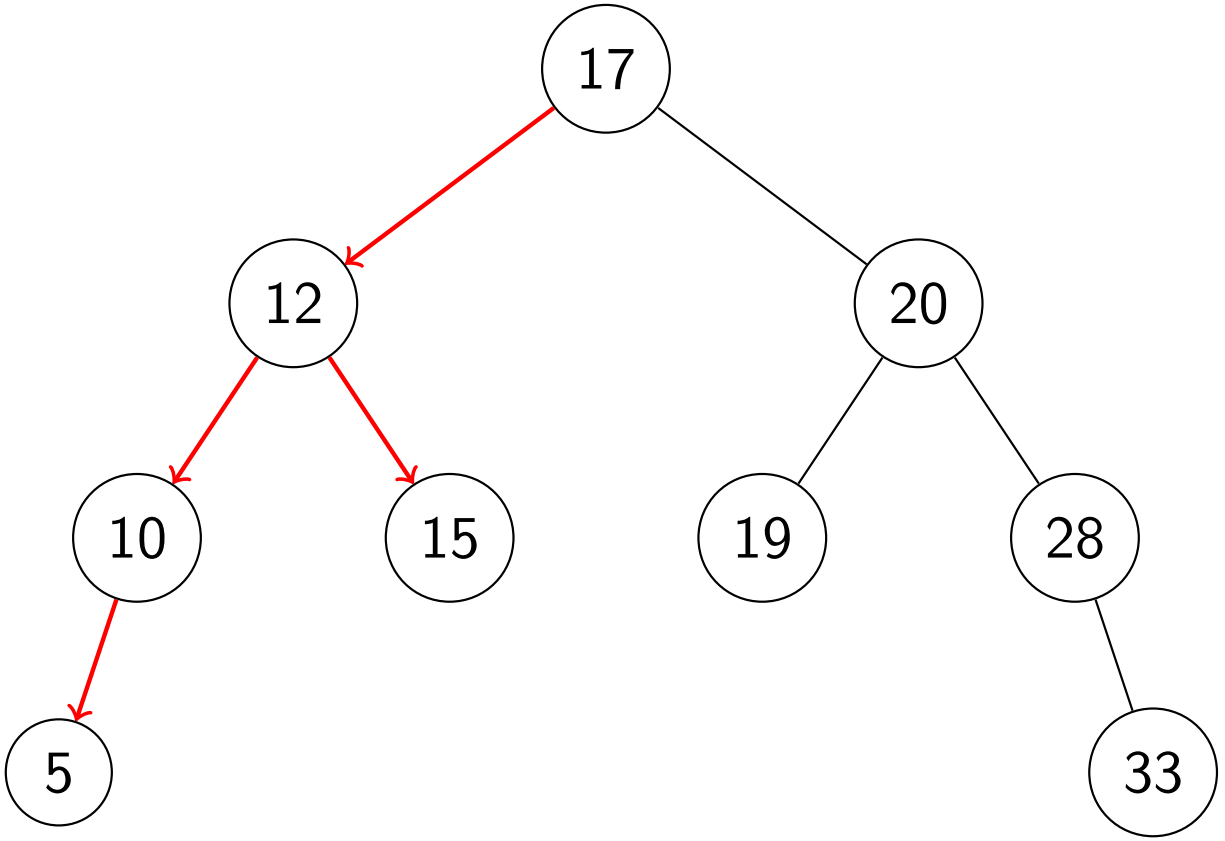


Output:

5
10
12

In-Order (left - output - right)

Tree:

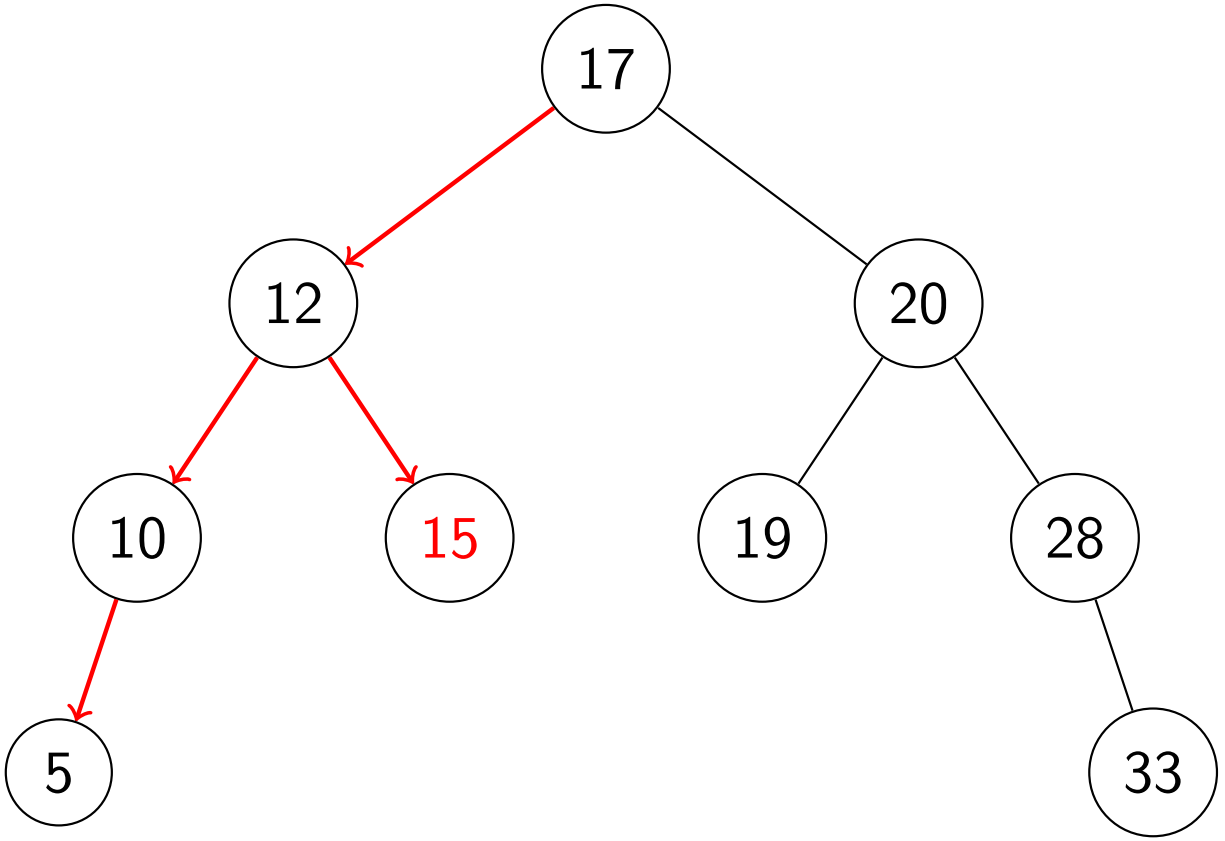


Output:

5
10
12

In-Order (left - output - right)

Tree:

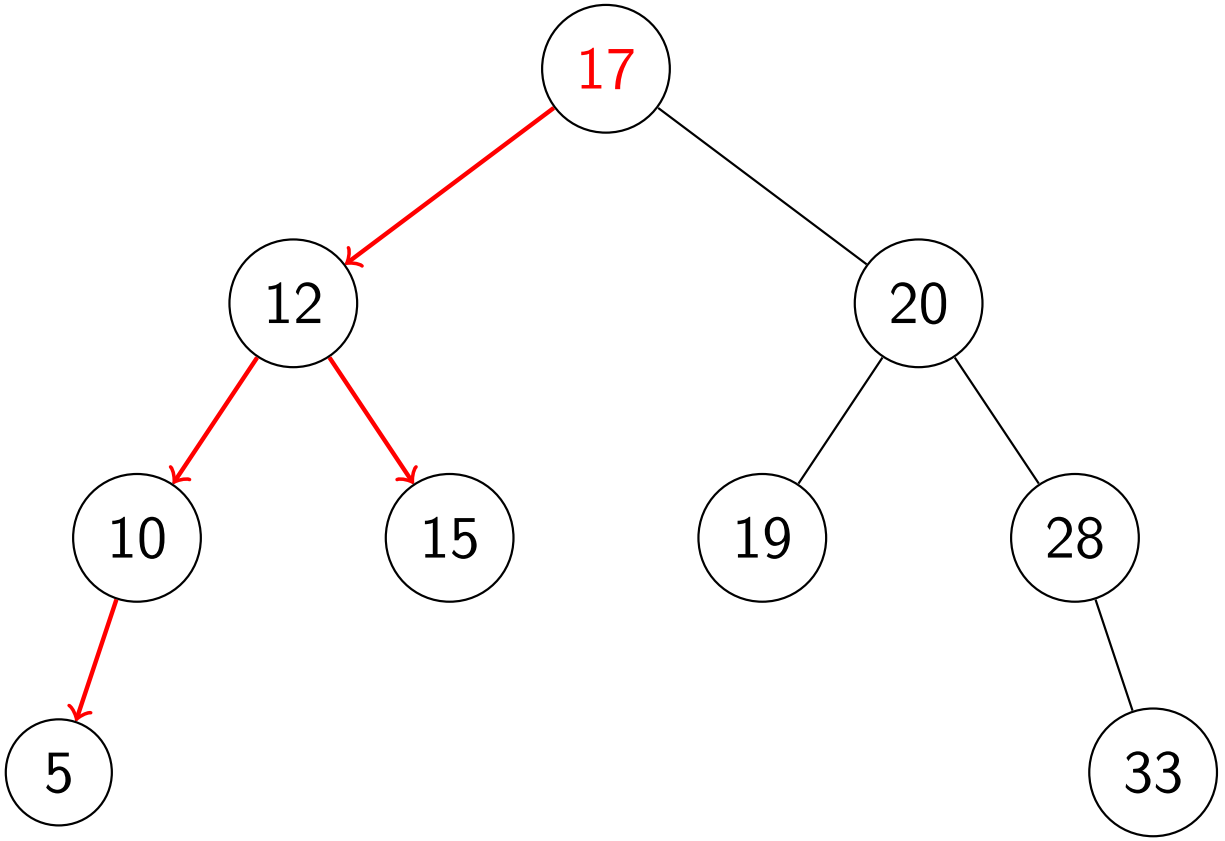


Output:

5
10
12
15

In-Order (left - output - right)

Tree:

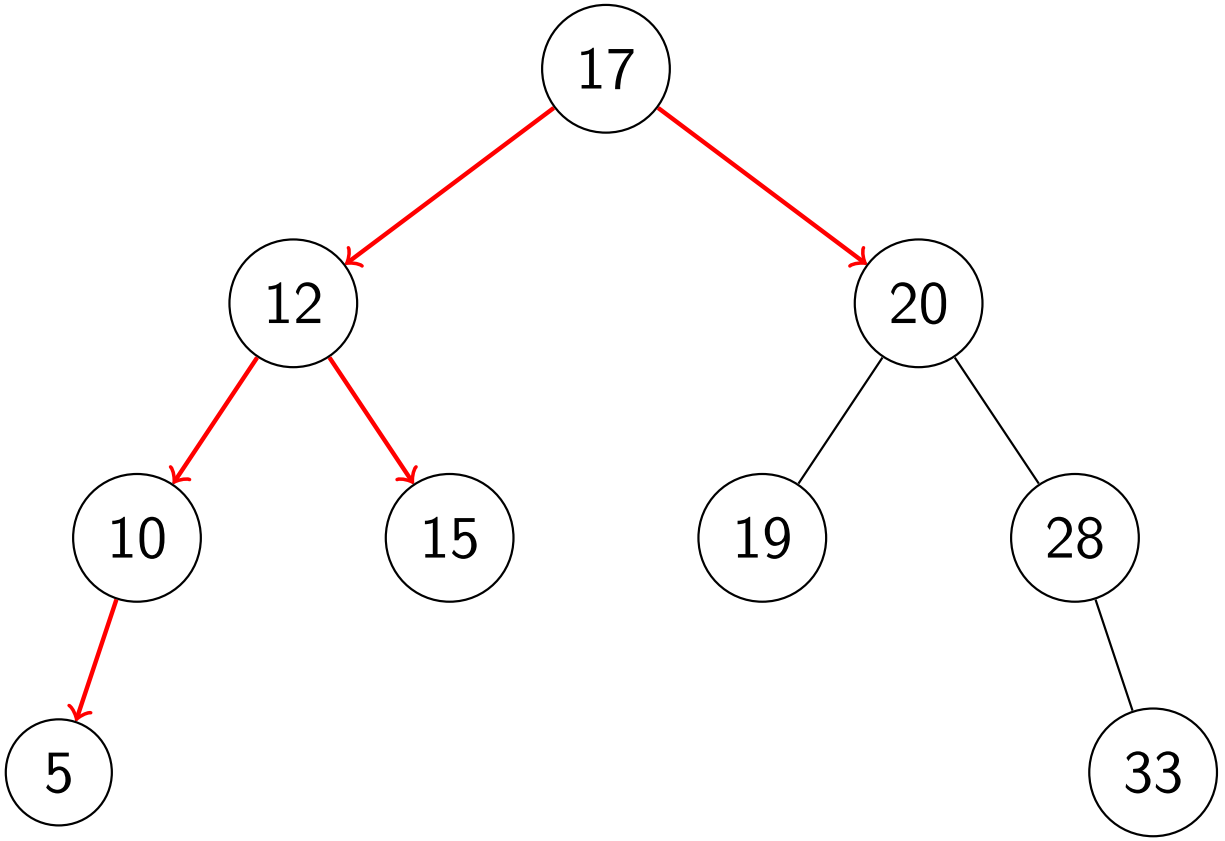


Output:

5
10
12
15
17

In-Order (left - output - right)

Tree:

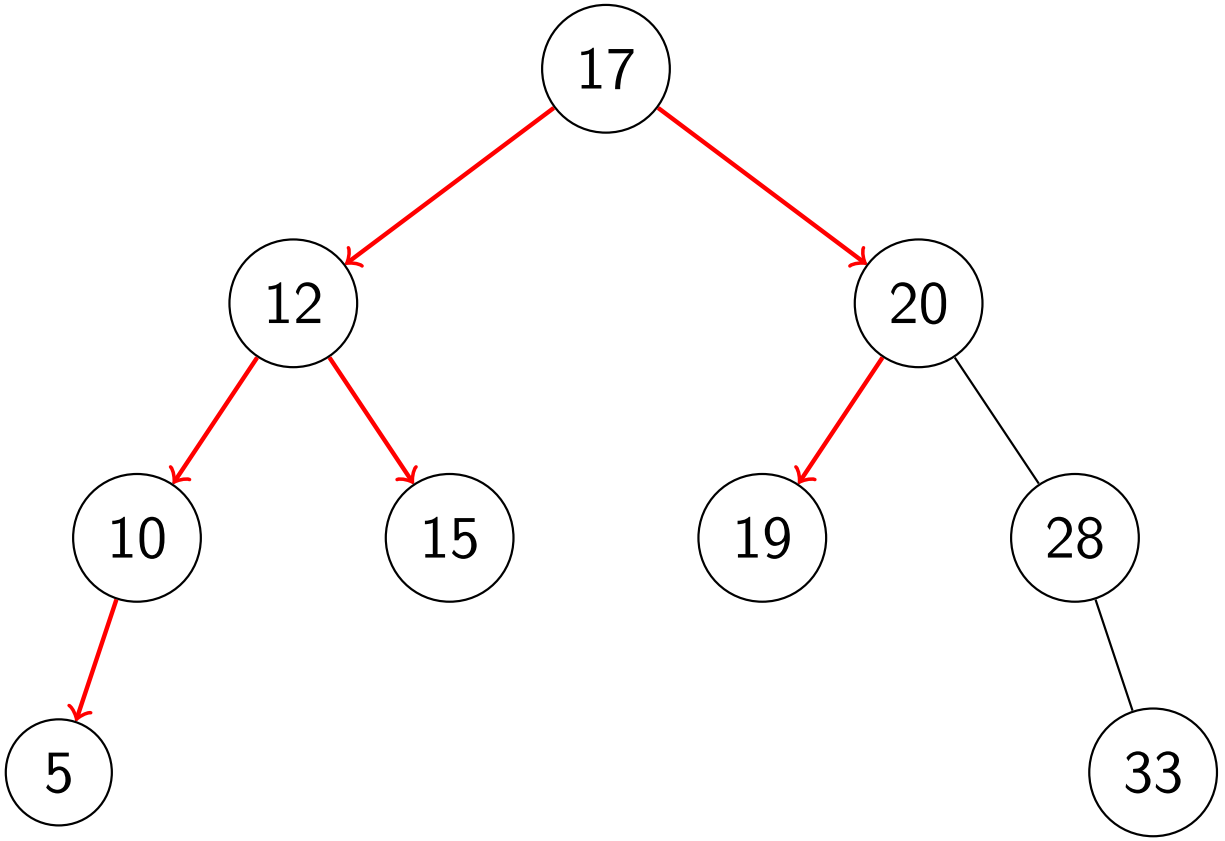


Output:

5
10
12
15
17

In-Order (left - output - right)

Tree:

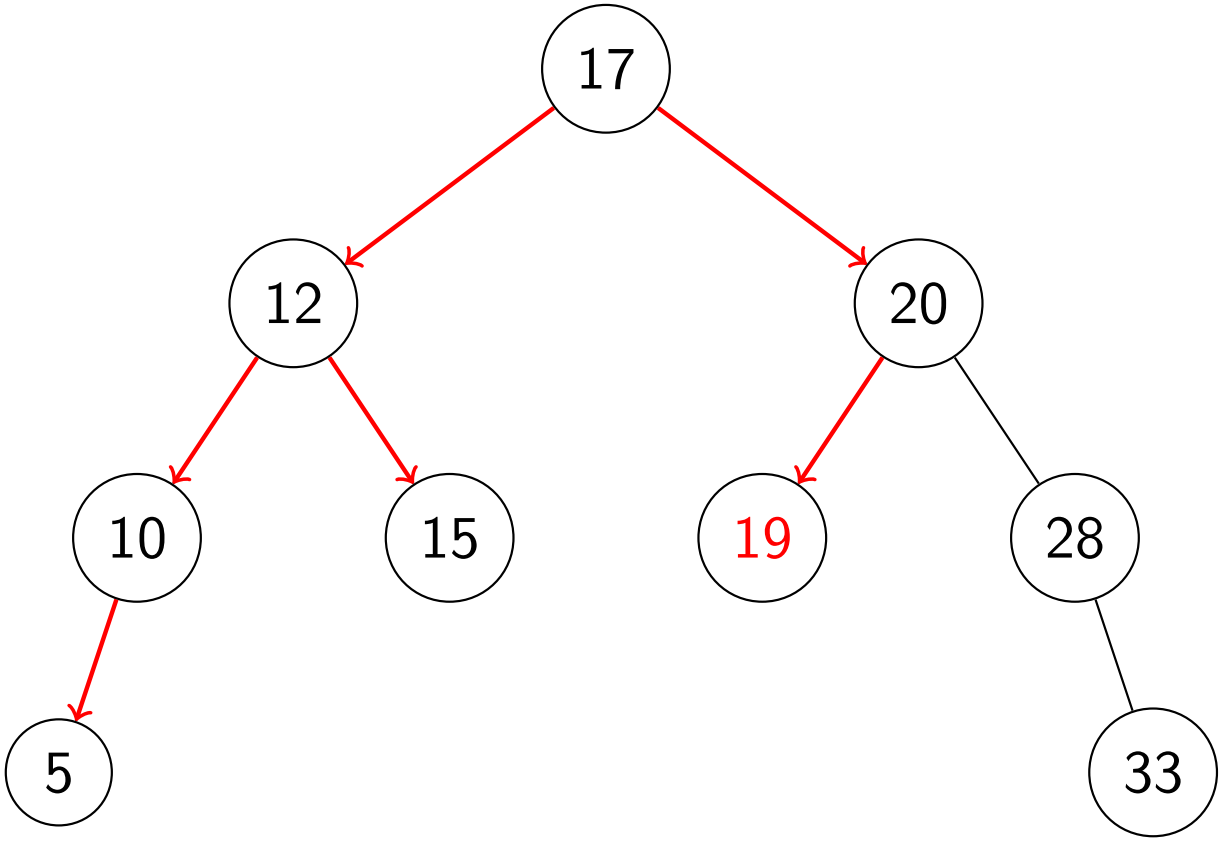


Output:

5
10
12
15
17

In-Order (left - output - right)

Tree:

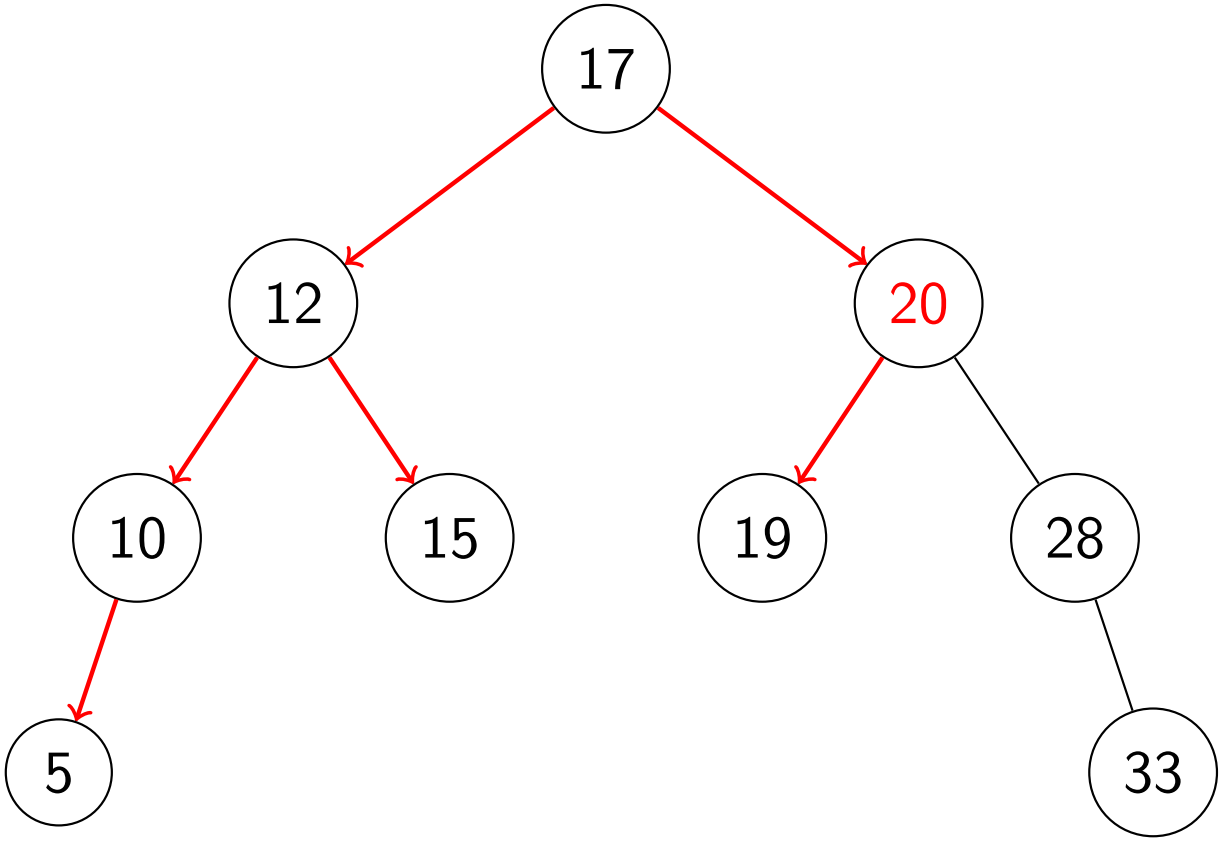


Output:

5
10
12
15
17
19

In-Order (left - output - right)

Tree:

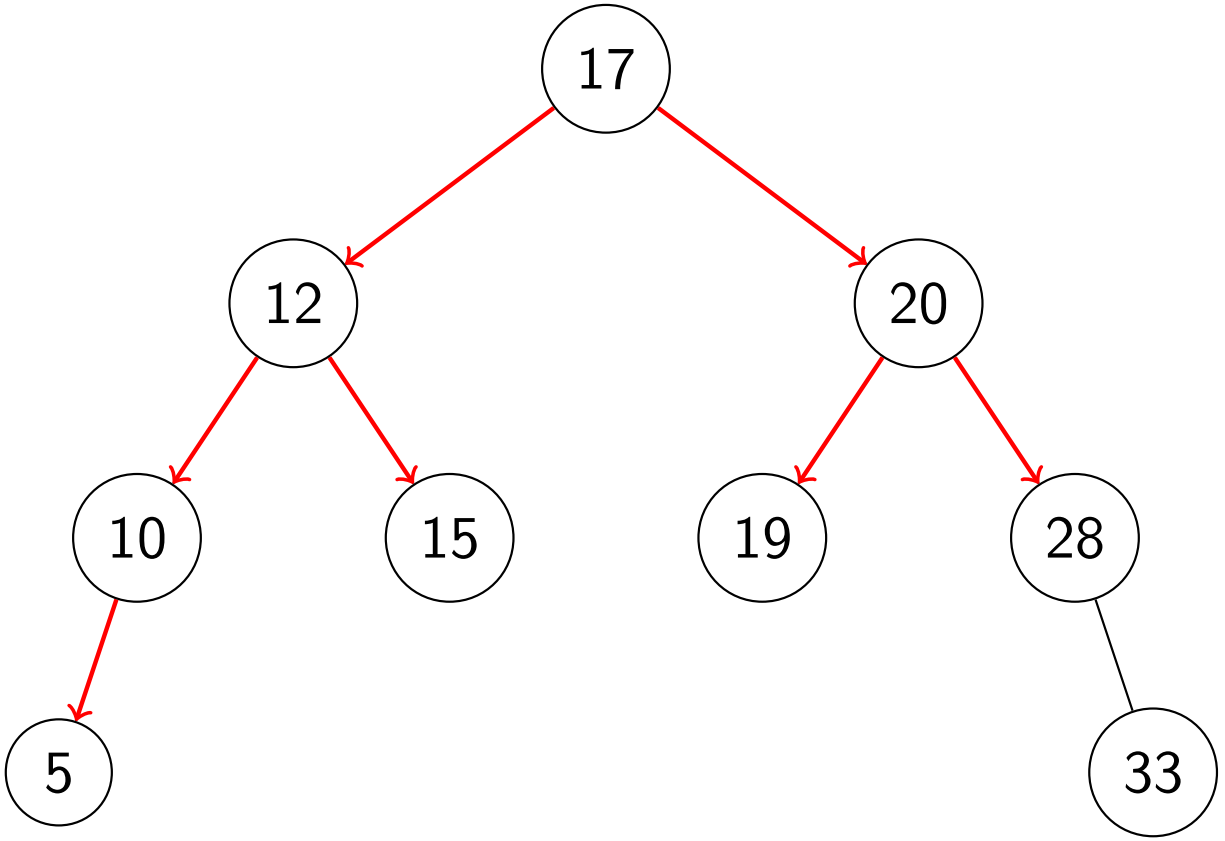


Output:

5
10
12
15
17
19
20

In-Order (left - output - right)

Tree:

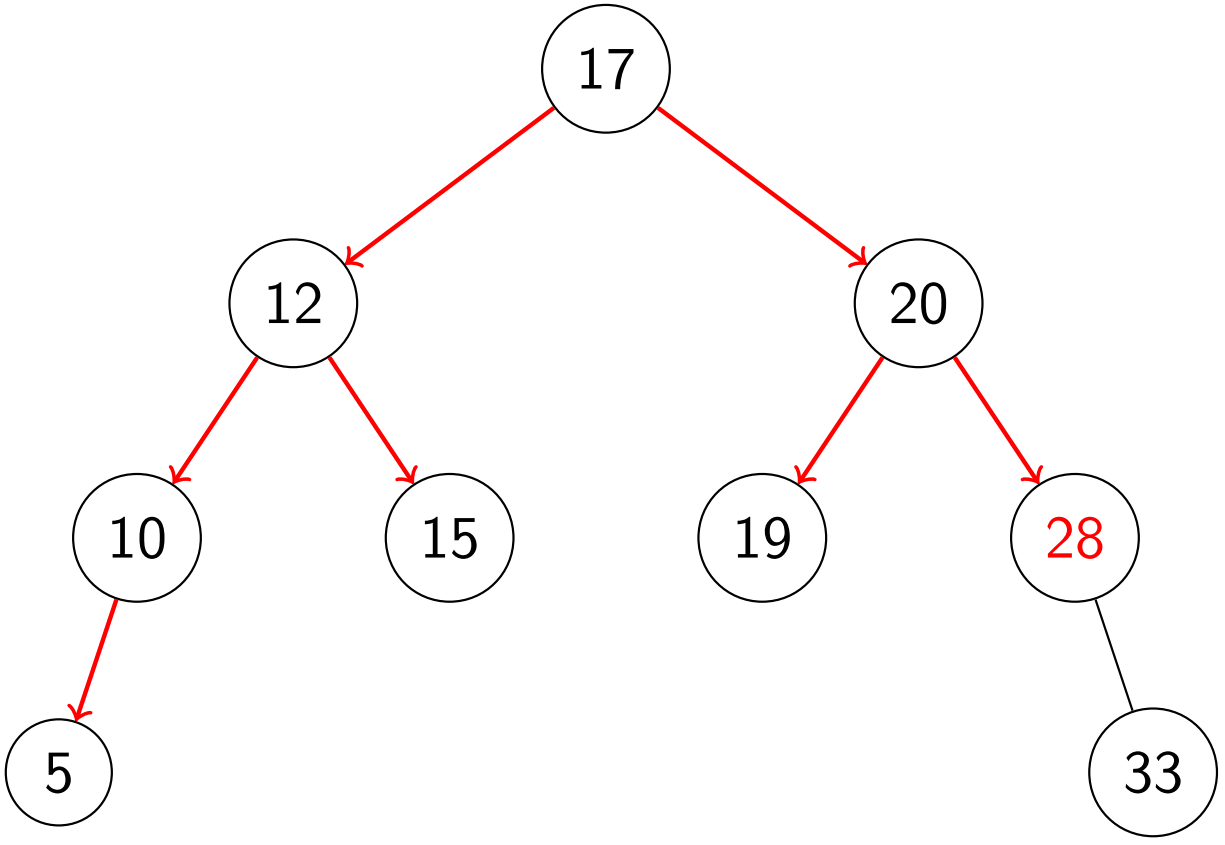


Output:

5
10
12
15
17
19
20

In-Order (left - output - right)

Tree:

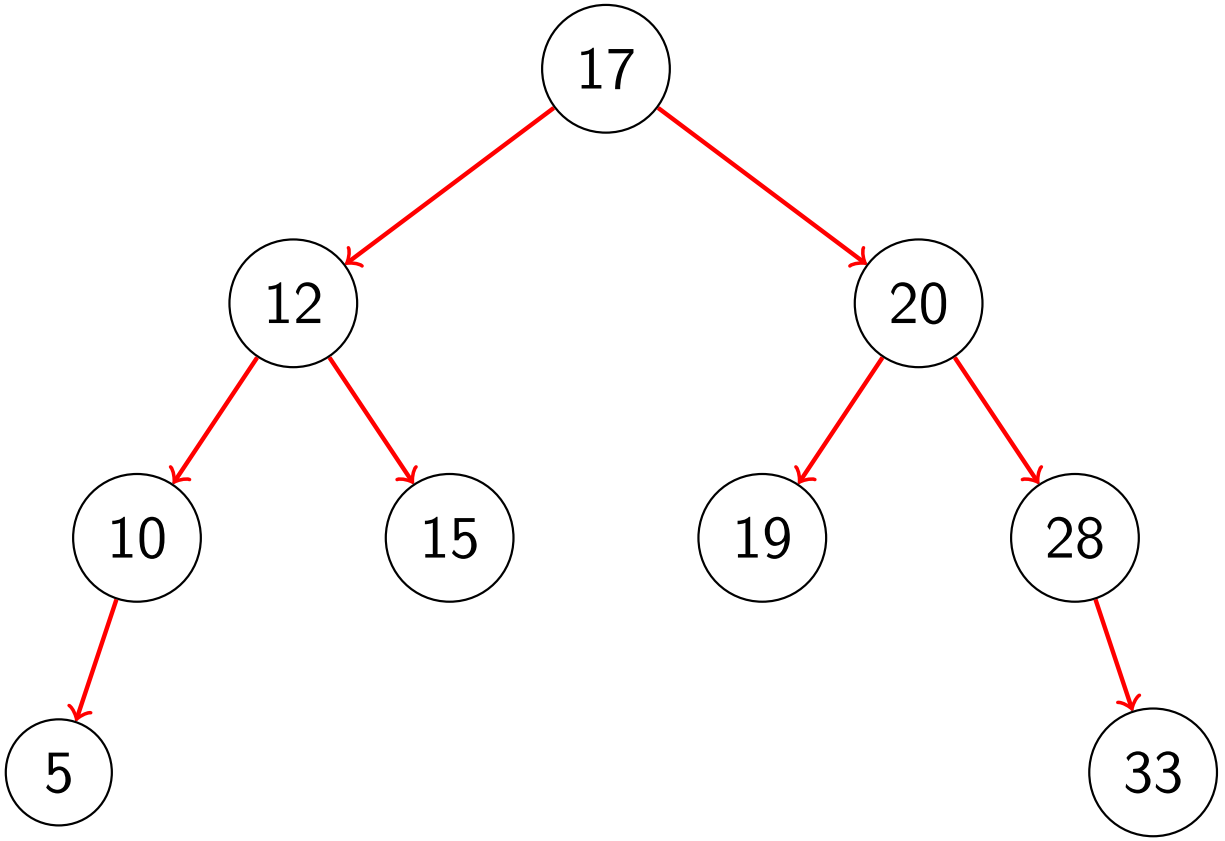


Output:

5
10
12
15
17
19
20
28

In-Order (left - output - right)

Tree:

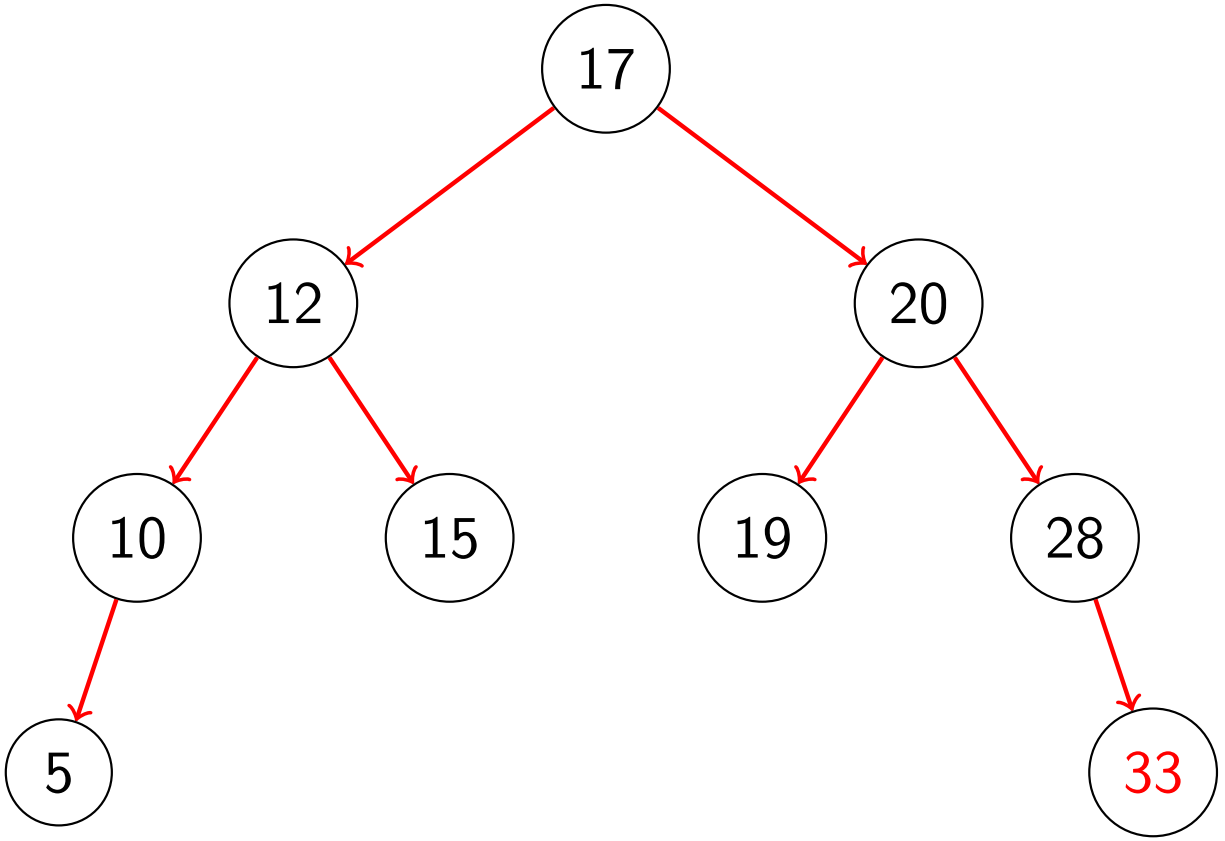


Output:

5
10
12
15
17
19
20
28

In-Order (left - output - right)

Tree:



Output:

5
10
12
15
17
19
20
28
33

Implementation

in R code

```
insert <- function(tree, data)
{
  if(is.null(tree$data))
    tree <- list(left=NULL, right=NULL, data=data)
  else if(data < tree$data)
    tree$left <- insert(tree$left, data)
  else
    tree$right <- insert(tree$right, data)
  return(tree)
}
```

```
pre_order <- function(tree)
{
  if(is.null(tree$data)) return()
  cat(tree$data, " ")
  pre_order(tree$left)
  pre_order(tree$right)
}

post_order <- function(tree)
{
  if(is.null(tree$data)) return()
  post_order(tree$left)
  post_order(tree$right)
  cat(tree$data, " ")
}

in_order <- function(tree)
{
  if(is.null(tree$data)) return()
  in_order(tree$left)
  cat(tree$data, " ")
  in_order(tree$right)
}
```

```
tree_minimum <- function(tree)
{
  while(!is.null(tree$left$data))
    tree <- tree$left
  return(tree$data)
}

tree_maximum <- function(tree)
{
  while(!is.null(tree$right$data))
    tree <- tree$right
  return(tree$data)
}
```

```
tree_minimum_recursive <- function(tree)
{
  if(is.null(tree$left$data)) return(tree$data)
  return(tree_minimum_recursive(tree$left))
}

tree_maximum_recursive <- function(tree)
{
  if(is.null(tree$right$data)) return(tree$data)
  return(tree_maximum_recursive(tree$right))
}
```

```
tree <- list()
for(i in c(17, 12, 10, 5, 15, 20, 19, 28, 33))
  tree <- insert(tree, i)

cat("Pre-Order:    ")
pre_order(tree)
cat("\nPost-Order:   ")
post_order(tree)
cat("\nIn-Order:     ")
in_order(tree)
cat("\n\n")
cat("Tree minimum: ", tree_minimum(tree), "(
    iterative)", tree_minimum_recursive(tree), "(
    recursive)\n")
cat("Tree maximum: ", tree_maximum(tree), "(
    iterative)", tree_maximum_recursive(tree), "(
    recursive)\n")
```

Script output

```
> source("ex39_walking_trees.R")
Pre-Order:    17  12  10  5  15  20  19  28  33
Post-Order:   5  10  15  12  19  33  28  20  17
In-Order:     5  10  12  15  17  19  20  28  33

Tree minimum:  5 (iterative) 5 (recursive)
Tree maximum:  33 (iterative) 33 (recursive)
```

in C++ code

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

node *new_node()
{
    node *n = (node *) malloc(sizeof(node));
    n->left = n->right = NULL;
    return n;
}

void insert(node **root, int data)
{
    if(!*root) *root = new_node();
    if(!(*root)->data) (*root)->data = data;
    else if(data < (*root)->data)
        insert(&(*root)->left, data);
    else
        insert(&(*root)->right, data);
}
```

in C++ code

```
void inorder(node *root)
{
    if(root==NULL) return;
    inorder(root->left);
    printf("%2d ", root->data);
    inorder(root->right);
}
void free_tree(node *root)
{
    if(root==NULL) return;
    free_tree(root->left);
    free_tree(root->right);
    free(root);
}

int main(int argc, char *argv[])
{
    node *root = NULL;

    int values[] = {17, 12, 10, 5, 15, 20, 19, 28, 33};
    for(unsigned int i = 0; i<sizeof(values)/sizeof(int); i++)
        insert(&root, values[i]);

    printf("\nIn-Order:  "); inorder(root); printf("\n");

    free_tree(root);
    return 0;
}
```