SYEDA NIMRA
USN: 1RV24MC009

DSA [SKILL LAB]

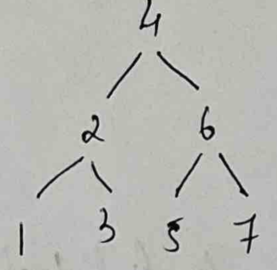
Finding Height of a binary Tree:

-> The length of the longest path from the root of a binary tree to a leaf node is the height of the binary tree.

-> The height of the root is the height of the tree

-> Height of tree = number of edges in the longest path connecting root to any leaf node.

Example 1:

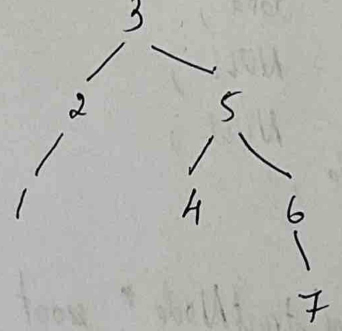


Height = 2

The number of edges from 4 to any leaf node in the above ~~exam~~ example is 2.

Therefore height of the tree is 2.

Example 2:



height = 3

The number of edges from root node, 3 to the last leaf node, 7 is 3.

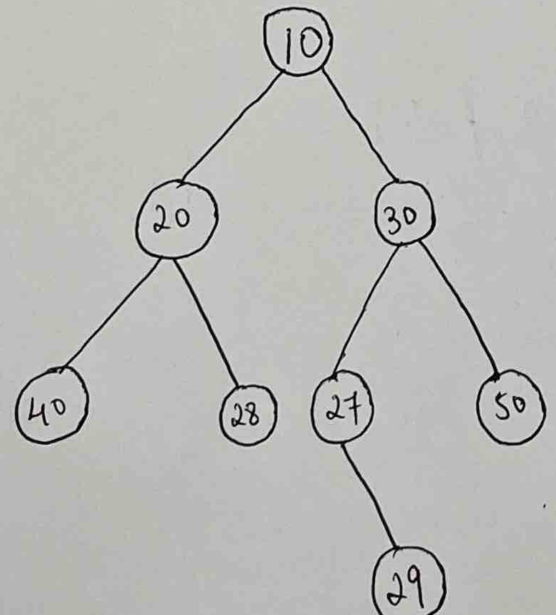
Therefore height of the tree is 3.

Code to find height of binary tree :

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

  struct Node{                         //structure of binary tree
       int data;
       struct Node* left;
       struct Node * right;
  };

  struct Node* newNode(int data){        //function to create a new node
    struct Node * node = (struct Node*) malloc(sizeof(struct Node));
    node -> data = data;
    node -> left = NULL;
    node -> right = NULL;
    return node;
  }

  int heightof Tree(structNode * root)        // function to find height
  {                                           of binary tree.
    if(root = =NULL)
      return -1;
   int leftHeight = heightof tree(root -> left);
   int rightHeight = height of tree (root -> right);
   return ( leftHeight > rightHeight? left Height : rightHeight)+1;

  }

  int main() {                               //Driver Code
   struct Node* root = newNode(10);          //binary tree formation
   root-> left = newNode(20);
root -> right = newNode (30);
root -> left -> left = newNode (40);
root -> left -> right = newNode (28);
root -> right -> left = newNode (27);
root -> right -> right = newNode (50);
root -> right -> left -> right = newNode (29);
printf ("Height of a given binary tree is %d \n", heightOfTree
                                                      (root));
return 0;
}
```

Output:

The ~~given~~ height of a given binary tree is 3.



Smallest number with atleast n trailing zeros in a factorial

Given a number n, the task is to find the smallest number whose factorial contains atleast n trailing zeros.

1) Input n=1 // we need to find the smallest number whose factorial contains atleast n trailing zeros.
Output 5

1! 2! 3! 4! does not contain a trailing zero

5! = 120 which contains one trailing zero.

2) Input n=6 // we need to find the smallest number whose factorial has atleast 6 trailing zeros.
Output: 25

Code Implementation:

```
#include <stdio.h>
int check (int p, int n) {
    int temp = p , count = 0, f = 5;
    while (f <= temp) {
        count += temp/f ;
        f *= 5 ;
    }
    return (count >= n);
}

int findNum(int n)
{
  if (n==1)
      return 5;

  int low = 0, high = 5 * n ;
while (low < high) {
  int mid = (low + high)/2 ;
  if (check (mid, n) )
      high = mid
  else
      low = mid + 1;
  }
  return low ;
}
 int main ()
 {
   int n=6 ;
   printf (" Smallest number with atleast %d trailing zeros
    in a factorial : %d\n ", n, findNum (n)) ;
    return 0 ;
 }
```

Output:

Smallest number with atleast 6 trailing zeros in a factorial is : 25.