```cpp
#include <iostream>
#include <cassert>

using namespace std;

class Node{
public:
  Node(int aKey):mKey(aKey),mpLeftChild(0),mpRightSibling(0){}
  Node* Clone(){return new Node(mKey);}
public:
  int mKey;
  Node* mpLeftChild;
  Node* mpRightSibling;
};

class Tree{
public:
  Tree();
  ~Tree();
  void OutputPostOrder(ostream& out)const;
  void Halve();
private:
  void mHalve(Node* apNodeCurrent, int aDepth);
  void mDistructor(Node* apNode);
  void mOutputPostOrder(Node* apNode,ostream& out)const;
private:
  Node* mpRoot;
};

void Tree::Halve()
{
  mHalve(mpRoot,0);
}


//La procedura halve dimezza l'altezza dell'albero rendendo i nodi di profondita' pari figli del padre del padre
void Tree::mHalve(Node* apNodeCurrent, int aDepth)
{
  if (apNodeCurrent!=0)
    {
      //recurr on all children
      Node* child_cursor=apNodeCurrent->mpLeftChild;
      while (child_cursor!=0)
    {
      mHalve(child_cursor,aDepth+1);
      child_cursor=child_cursor->mpRightSibling;
    }
      //if depth is odd then process children
      if (aDepth%2!=0)
    {
      //append left child to last right sibling
      Node* cursor=apNodeCurrent->mpLeftChild;
      while (cursor->mpRightSibling!=0) cursor=cursor->mpRightSibling;
      cursor->mpRightSibling=apNodeCurrent->mpLeftChild;
      //remove link to children
      apNodeCurrent->mpLeftChild=0;
    }
```

```
    }
}

Tree::Tree():mpRoot(0){}


Tree::~Tree()
{
  mDistructor(mpRoot);
}

void Tree::mDistructor(Node* apNode)
{
  if (apNode !=0)
    {
      mDistructor(apNode->mpLeftChild);
      mDistructor(apNode->mpRightSibling);
      delete apNode;
    }
}


void Tree::OutputPostOrder(ostream& out)const
{
  mOutputPostOrder(mpRoot,out);
}

void Tree::mOutputPostOrder(Node* apNode, ostream& out)const
{
  if (apNode!=0)
    {
      Node* cursor=apNode->mpLeftChild;
      while (cursor!=0)
    {
      mOutputPostOrder(cursor, out);
      cursor=cursor->mpRightSibling;
    }
      out<<apNode->mKey<<" ";
    }
}

ostream& operator<<(ostream& out, const Tree& aTree)
{
  aTree.OutputPostOrder(out);
  return out;
}
```