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

using namespace std;

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

class Tree{
public:
  Tree();
  ~Tree();
  void OutputPreOrder(ostream& out)const;
  void Input(const int* aSequence, const int aSize);
private:
  void mInput(Node* aNode, const int* aSequence, const int aSize, int& aIndex);
  void mDistructor(Node* apNode);
  void mOutputPreOrder(Node* apNode,ostream& out)const;
private:
  Node* mpRoot;
};


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


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

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


void Tree::OutputPreOrder(ostream& out)const
{
  mOutputPreOrder(mpRoot,out);
}

void Tree::mOutputPreOrder(Node* apNode, ostream& out)const
{
  if (apNode!=0)
```

```
    {
      out<<apNode->mKey<<" ";
      mOutputPreOrder(apNode->mpLeftChild, out);
      mOutputPreOrder(apNode->mpRightChild, out);
    }
}

void Tree::Input(const int* aSequence, const int aSize)
{
  if (aSize>0)
    {
      int index=0;
      mpRoot=new Node(aSequence[index]);
      mInput(mpRoot,aSequence,aSize,index);
    }
}

//Reads the input sequence and builds a binary tree considering:
//1. odd elements are leaf nodes
//2. even elements are internal nodes
//3. the sequence is given in pre-order
void Tree::mInput(Node* aNode, const int* aSequence, const int aSize, int& aIndex)
{
  ++aIndex;
  if (aIndex<aSize)
    {
      aNode->mpLeftChild=new Node(aSequence[aIndex]);
      if (aSequence[aIndex]%2==0)
    mInput(aNode->mpLeftChild,aSequence,aSize,aIndex);
    }
  ++aIndex;
  if (aIndex<aSize)
    {
      aNode->mpRightChild=new Node(aSequence[aIndex]);
      if (aSequence[aIndex]%2==0)
    mInput(aNode->mpRightChild,aSequence,aSize,aIndex);
    }
}

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

int main()
{
  int sequence[]={6,2,1,8,1,1,4,5};
  int size=8;
  Tree t;
  t.Input(sequence,size);
  cout<<t<<endl;

  return 0;
}
```