MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

MASTER COPY
PISA

Esercizio

Scrivere un programma che legga da tastiera un intero N, una sequenza A di N coppie [**intero,stringa**] ed un intero D. Ogni coppia rappresenta il nodo di un albero binario di ricerca. Il programma deve:

- Inserire uno alla volta, nell'ordine dato, le coppie [**intero,stringa**] di A in un albero binario di ricerca **senza ribilanciamento**. L'inserimento deve essere tale per cui, per un qualsiasi nodo, il sottalbero sinistro contenga le coppie il cui valore intero è **minore o uguale** del valore intero del nodo, mentre il sottoalbero destro contiene le coppie il cui valore intero è **maggiore**.
- Stampare **in ordine lessicografico** tutte le stringhe che si trovano nell'albero in nodi a profondità D. Si ricorda che la profondità di un nodo è uguale alla sua **distanza dalla radice e che la profondità della radice è 0**

L'input è formattato nel seguente modo. La prime due righe contengono i due interi N e D. Seguono $2N$ righe, due righe per coppia. Per ogni coppia abbiamo la prima riga che contiene il valore intero e la seconda che contiene la stringa associata. Si assume che N è sempre maggiore di zero.

L'output, se esistono nodi a profondità D, deve stampare nella prima riga il numero di nodi trovati e successivamente una riga per ogni stringa trovata. **Si ricorda che le stringhe vanno restituite in ordine lessicografico.** Se non esistono nodi a profondità D, il programma stampa solo 0.

MASTER COPY
PISA

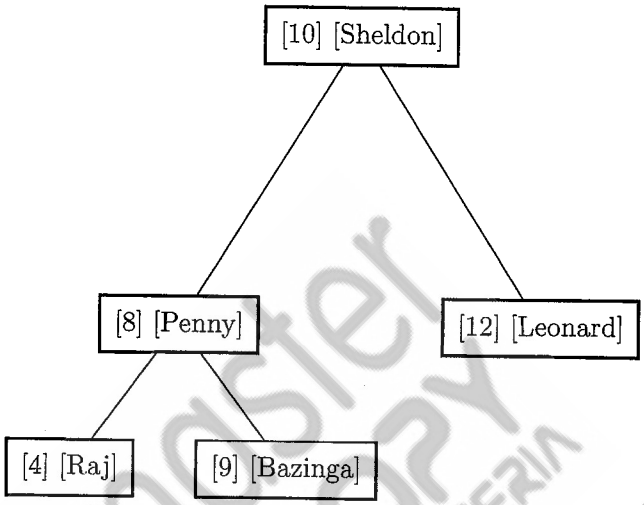
MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

Esempi

Input

```
5
2
10
Sheldon
12
Leonard
8
Penny
4
Raj
9
Bazinga
```

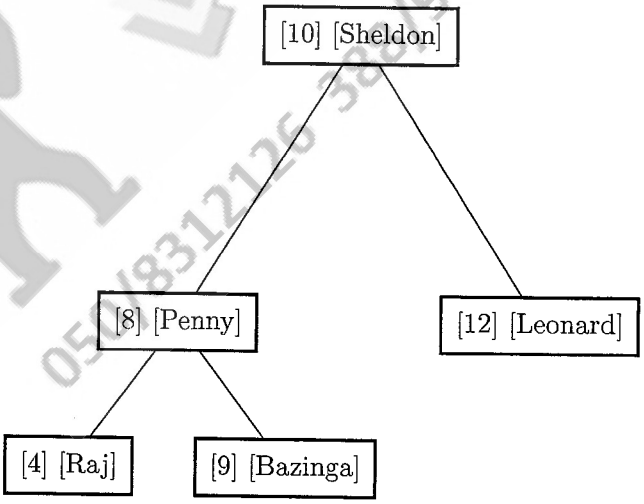


Output

```
2
Bazinga
Raj
```

Input

```
5
10
10
Sheldon
12
Leonard
8
Penny
4
Raj
9
Bazinga
```



Output

```
0
```

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

```
#include <vector>
#include <string>
#include <cstdlib>
#include <algorithm>
#include <iostream>
using namespace std;

struct Node
{
 int key_;
 string string_;
 int d;
 Node*left;
 Node*right;
 Node(int k,string s):
  key_(k),string_(s),left(NULL),right(NULL) {}
};

vector<string> V;

bool ordina( string s1 ,string s2)
{
 if(s1 < s2)
  return true;
 else
  return false;
};

class BinTree
{
  Node*root_;
 public:
  BinTree(){ root_ = NULL; };
  Node*getRoot(){ return root_; };
  void insert(int k,string s);
  void inOrder(Node*tree);
  void salva(Node*tree, int D);
};

void BinTree::insert(int k,string s)
{
  Node*node = new Node(k,s);
  Node*pre = NULL;
  Node*post = root_;
  node->d = 0;
  while(post!=NULL)
  {
    pre = post;
    if(k<=post->key_)
```

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

master COPY COPISTERIA 050/8312126 388/9837745

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

```
    {
     node->d++;
     post = post->left;
    }
    else
    {
     node->d++;
     post = post->right;
    }
   }
   if( pre == NULL)
    root_ = node;
   else if( k <= pre->key_)
    pre->left = node;
   else
    pre->right = node;
};

void BinTree::inOrder(Node*tree)
{
  if(tree!=NULL)
  {
    inOrder(tree->left);
    cout<<tree->key_<<" "<<tree->string_<<" "<<tree->d<<endl;
    inOrder(tree->right);
  }
  cout<<endl;
};

void BinTree::salva(Node*tree,int D)
{
 if(tree!=NULL)
 {
   salva(tree->left,D);
   if(tree->d == D)
    V.push_back(tree->string_);
   salva(tree->right,D);
 }
};

int main()
{
 int N,k,D;
 string s;
 BinTree B;
 cin>>N>>D;
 for(int i=0;i<N;++i)
 {
   cin>>k>>s;
```

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

master COPY COPISTERIA 050/8312126 388/9837745

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745

```
  B.insert(k,s);
 }
 B.salva(B.getRoot(),D);
 if(V.empty()!=0)
  cout<<"0"<<endl;
 else
 {
   cout<<V.size()<<endl;
   sort(V.begin(),V.end(),ordina);
   for(int i=0;i<V.size();i++)
    cout<<V[i]<<endl;
 }

 return 0;
};
```

MASTER COPY
Tel. 050 8312126
Cell. 388 9837745



MASTER COPY
Tel. 050 8312126
Cell. 388 9837745