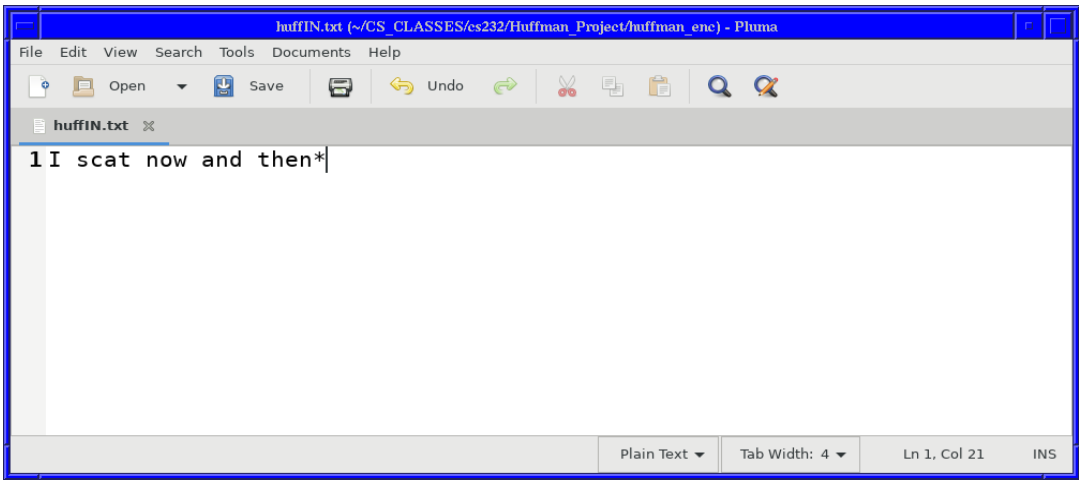
Consider the two phrases

i scat now and then the cats down in an

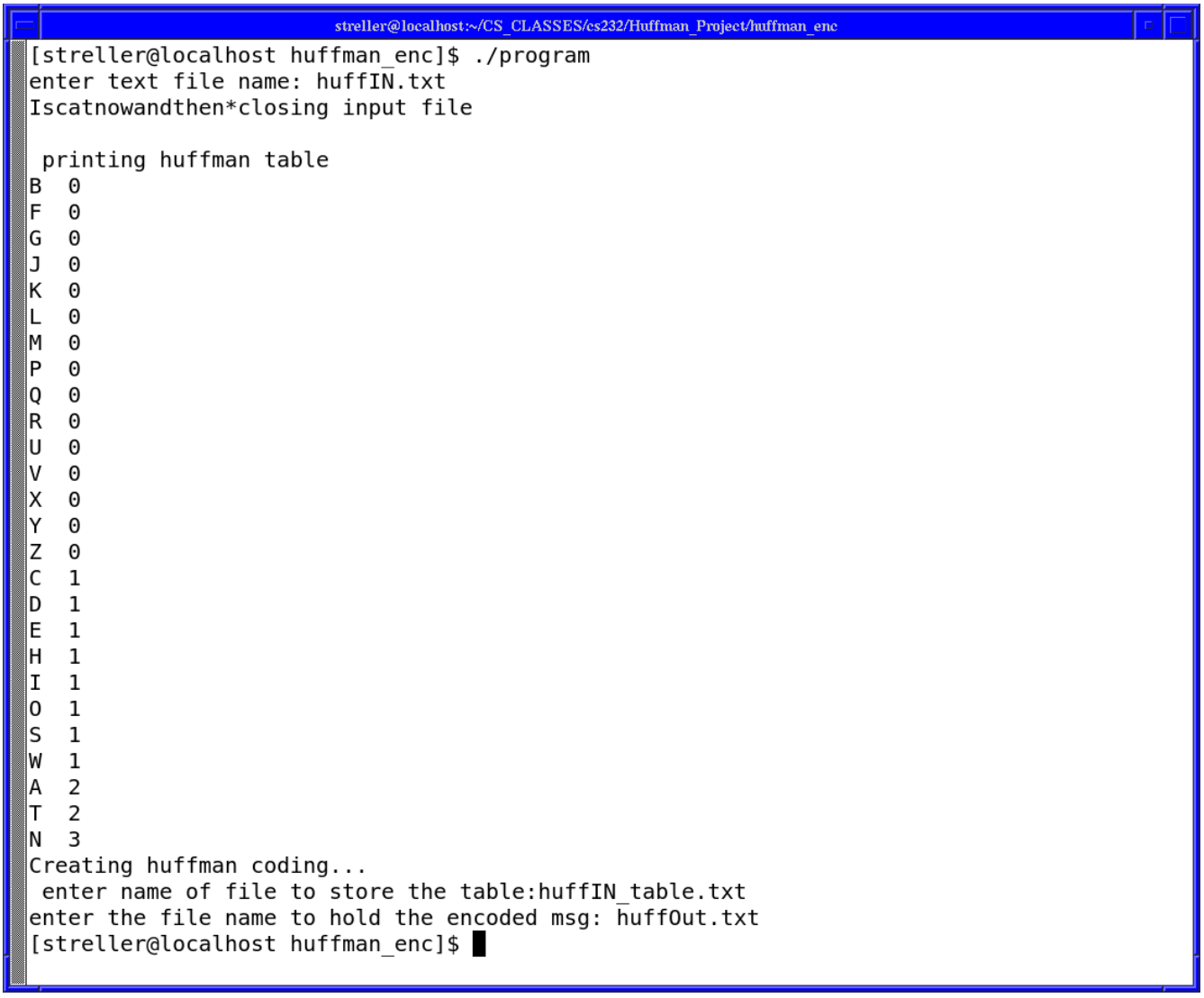
notice they have the same letters and frequency of letters. (we of course ignore spaces and case)

Your program will ask for a file by name (a txt file) which will contain a phrase with a terminating char of *
For example

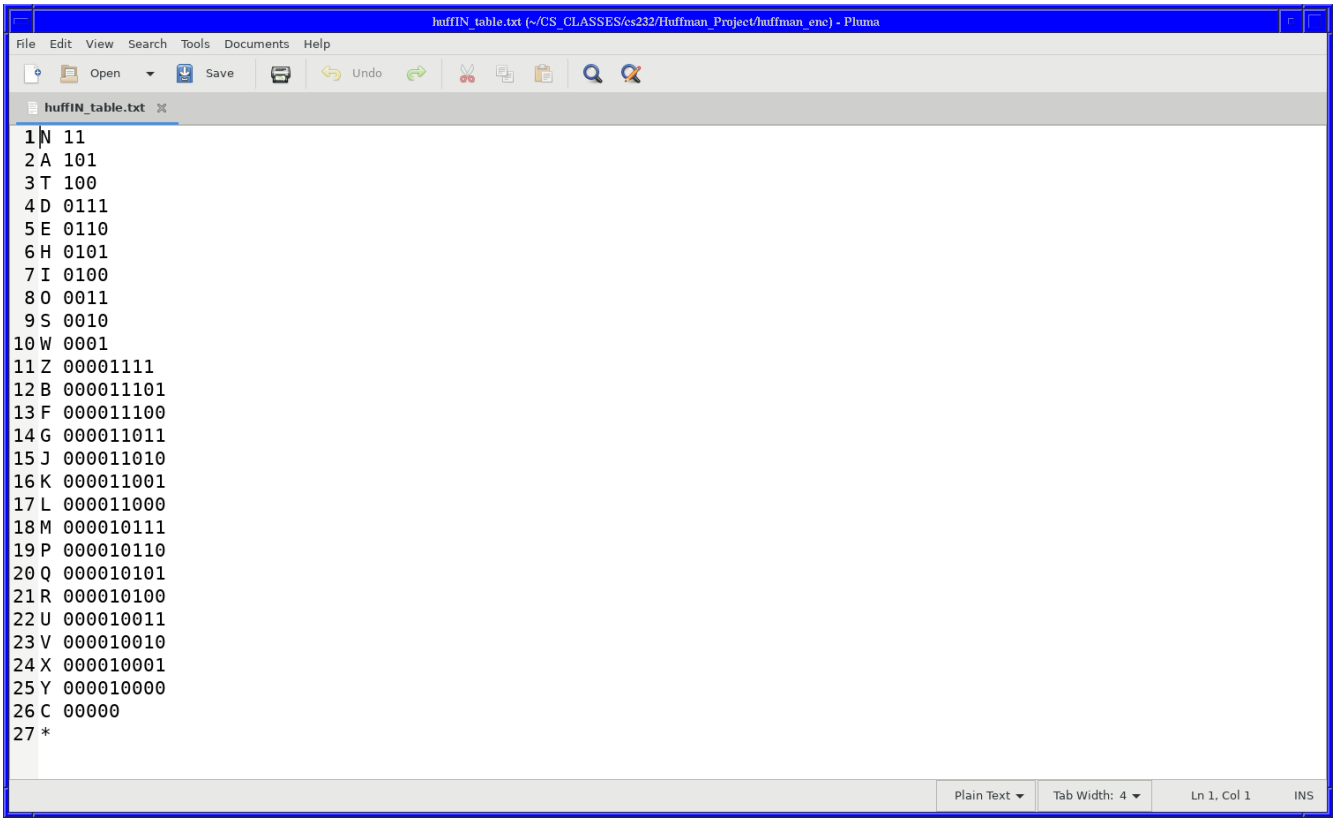


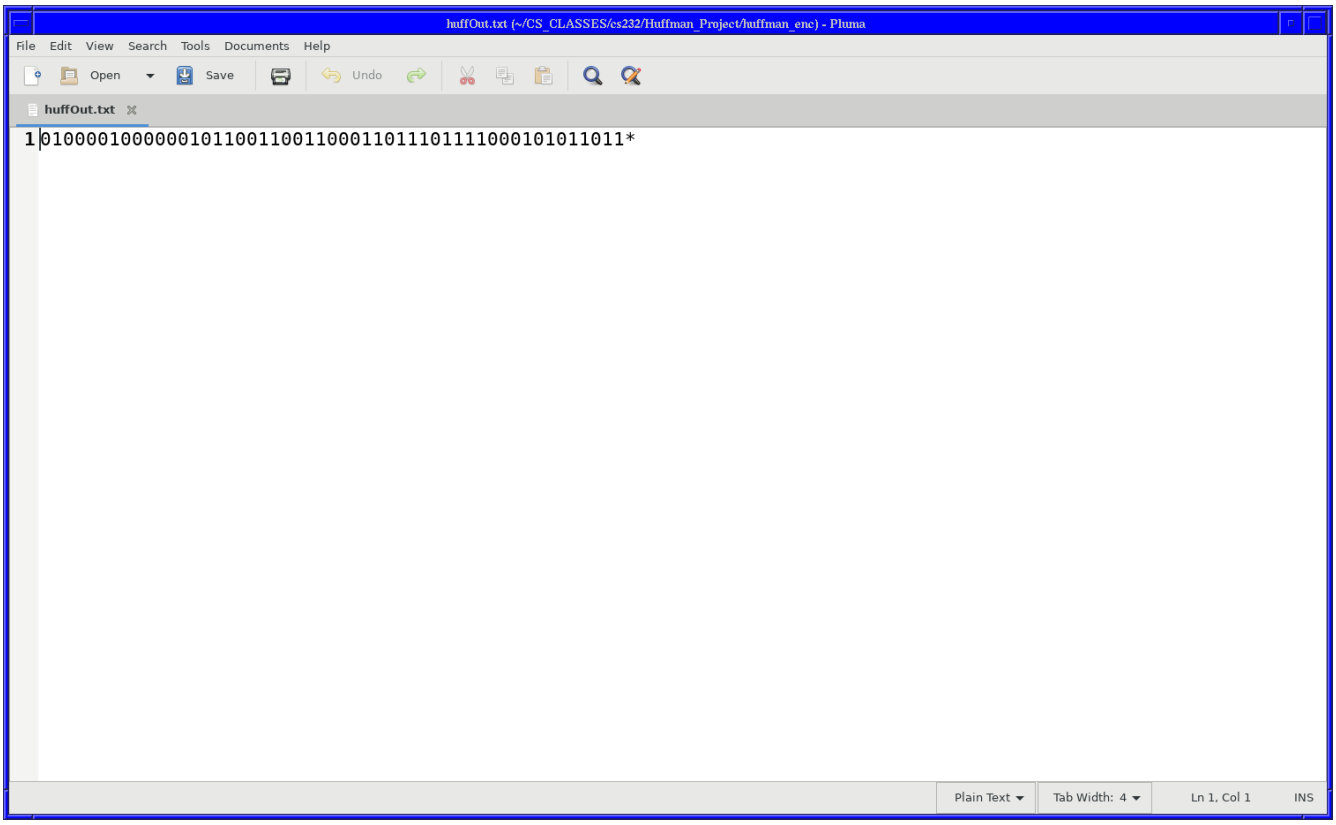
Your program will read this file, then create create the Huffman table which is to be displayed to the console and to a file (users creates name of the table file – its a txt of course.

Console output



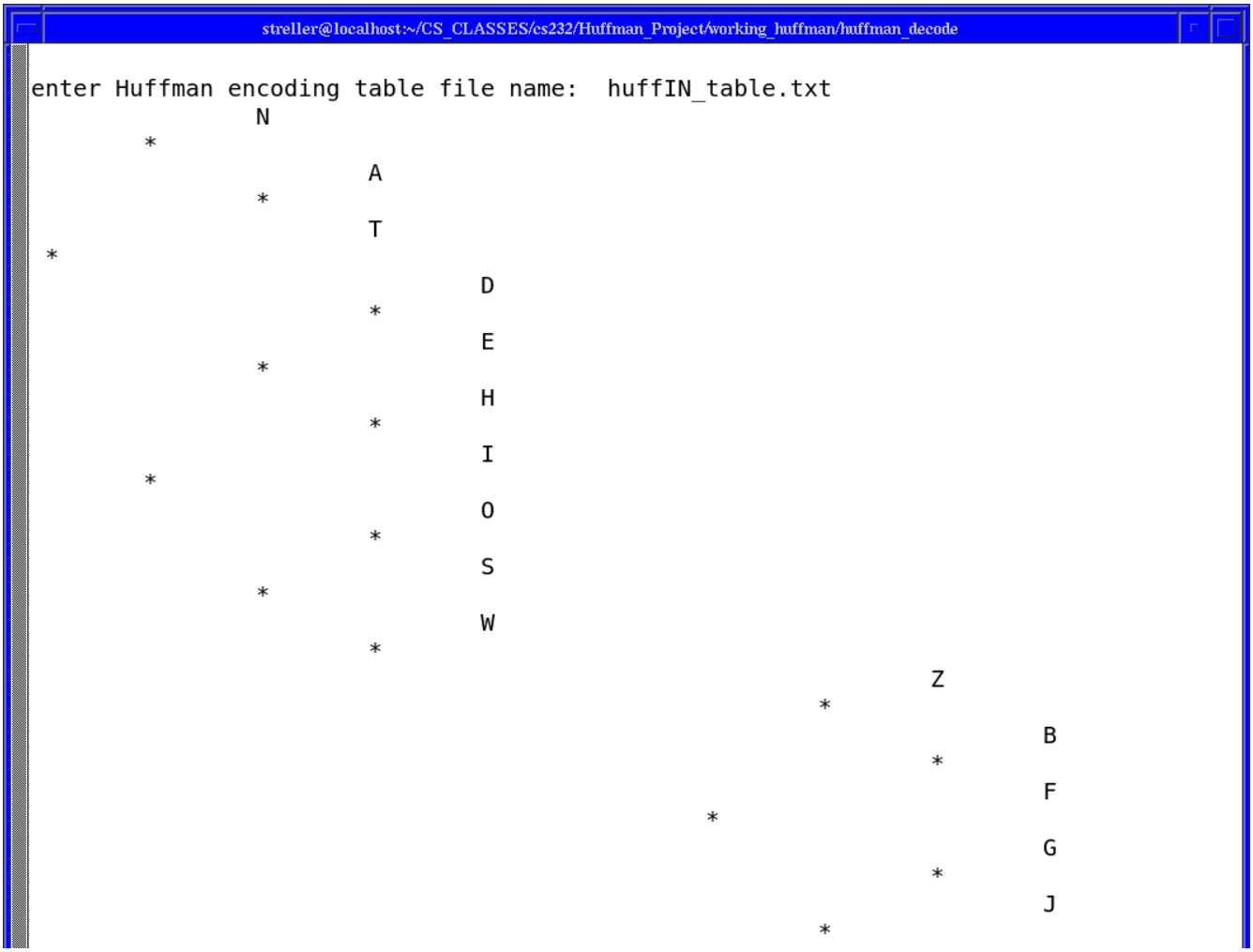
the files huffIN_table and huffOut

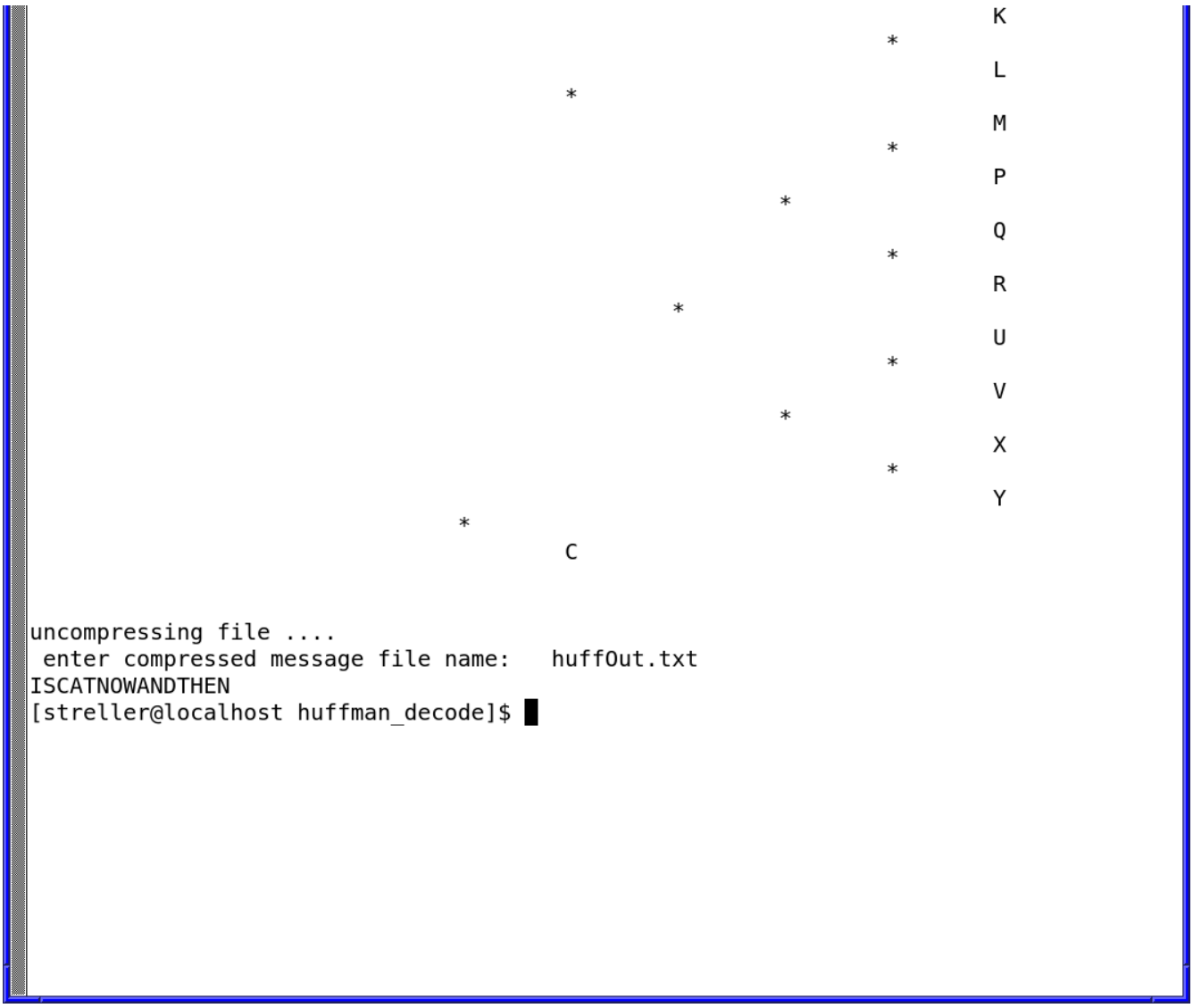




Thus you have created the encoded message .

Now your program must be able to decode a message: namely given a huffman table and a 01 encoded file it must decode back to english





Now the fun part: how do I know that your program really works ? After all this is just going in circles. Start with “i scat now and then” and end with ISCATNOWANDTHEN big deal still the same! So you eliminated white space and toUppered everything.

That’s where the other phrase “the cats down in an” comes into play.
As mentioned it has the same letters and same frequency of letters.

So using the previously created table huffIN_table.txt, create by hand a 01 file for this phrase.

HuffCat.txt
1000101011000000101100001001110011000111010011101111*

and have your program decode this file using the previously created huffIN_table.txt and the by hand created file HuffCat.txt

```
[streller@localhost huffman_decode]$ ./program
enter Huffman encoding table file name: huffIN_table.txt

uncompressing file ....
enter compressed message file name: huffCat.txt
THECATSDOWNINAN
[streller@localhost huffman_decode]$
```

Happy Happy Joy Joy.

Yes I dealt with everything in just one case (upper) that way my table is always 26 and not 52. You will only get alpha files.

Yes I created a tree with every one of the 26 letters. You can certainly just do tables/trees for the symbols that are actually in the file. (my approach is that the table/ list is always the same size)

Hints or How I did it (which may or may not help)

created a struct

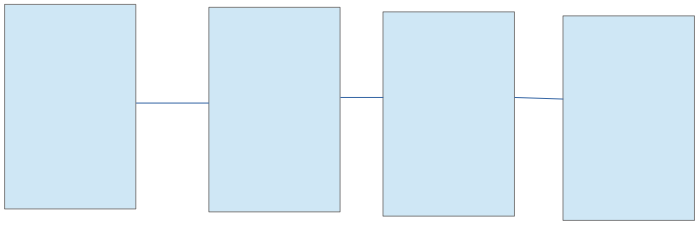
```
struct info
{
  char letter;
  int weight;
  info * llink, *rlink;
  bool operator < (info & n)
  {
    return weight < n.weight;
  }
```

(obviously holds a data element)

create a stl list of info to hold ALL data (why stl – cuz its got a sort and a pushback and popfront) and if you really think about it, a linked list, with appropriate node struct is really tree (or a tree is just a linked list

But be careful : with the stl have we ever talked bout the link field? No. we just use iterators.

So what was done was to create stl list<info> (llink, rlink were set = 0)



where each node is

rlink
info
llink

So this creates a sorted stl linked list of the data.

Now you start making the tree;
popfront twice, (so you have the first 2 data element) create new info node, **now set values** to llink and rlink, push the new node back into list, sort and repeat (grab next 2, create new node….)