1.Is the tree that is created from the node files (dict * .dat) complete or almost complete? Justify your answer.

To know if a tree is complete or almost complete we have to look at its leaves , if the tree is complete , all its nodes will have either 0 or 2 sons, being the ones with 0 sons restricted to be in the last level otherwise the three will be almost complete or incomplete.
We can calculate if a tree is complete or not making simple operation , if $(2^{depth+1} -1)$ is equal to the number of nodes of the tree we can say that that tree is complete, else if $(2^{depth} -1)$<number of nodes<$(2^{depth+1} -1)$ we can know that the tree is going to be almost complete, otherwise the tree will be incomplete.
So if we have the depth of the tree and the number of nodes we can know if a tree is complete or almost complete.
Now let see this with all the .dat files:

- Dict1k.dat: the number of nodes is 1000 and the depth is 23 so this tree is incomplete because $(2^{23+1}-1)$> 1000 so that means there aren´t enough nodes to complete all the levels of the tree.
- Dict1M.dat: their number of nodes is 10000 and their depth is 48 so this tree is incomplete because $(2^{48+1}-1)$> 10000 so that means there aren´t enough nodes to complete all the levels of the tree.
- Dict10.dat: their number of nodes is 10 and their depth is 5 so this tree is incomplete because $(2^{5+1}-1)$> 10 so that means there aren´t enough nodes to complete all the levels of the tree.

2.

a) What is the relationship between the "shape" of a tree and its traversing modes?

This is a direct relationship, depending on the shape of the tree the different traversing modes will end up in a different way.

There are 3 modes of traversing a tree: pre-order, in-order and post-order. And the steps to execute each one, beginning on the root, are the following:

- Pre-order:
  Visit the vertex, Recursively traverse in pre-order the left child, Recursively traverse in pre-order the right child.
- In-order:
  Recursively traverse in-order the left child, Visit the vertex, Recursively traverse in-order the right child.
- Post-order:
  Recursively traverse in post-order the left child, Recursively traverse in post-order the right child, Visit the vertex.

b) Can you tell if a binary search tree is well constructed according to how it is traversed?

Yes you can, the easiest way to tell if it is well constructed is by traversing the tree in an in-order way, the result that you must get is that the elements are visited sorted. For example, in a BST of integers, on the result of an in-order traversing, all the integers that compose the tree should appear sorted.

3. Compare and describe the differences between the trees generated by the executables p4_e3 with the last argument B or N (number of nodes, depth, routes, etc.).

p4_e3 dict10.dat B

- Number of nodes=10
- Depth=3

p4_e2 dict10.dat N

- Number of nodes=10
- Depth=5

The principal difference of the two trees generated is the depth, the Balanced one has a depth of 3 and the other one a depth of 5. This is because when we introduce the flag B as argument, the generated tree is a more balanced tree, this is done by sorting the data of the file before inserting it on the tree. On the other hand, when you introduce the flag N as argument, the tree is generated by directly introducing the nodes has they are read form the file, therefore, unless they are already sorted on the file, the generated tree is much more likely to by deeper than the one generated with the flag B.

If you use N flag, higher amount of nodes usually means much higher depth, this can be seen by executing the program with the other dict*.dat files.