Homework 6 due 11/07/03

Hand in typed or handwritten C++ code containing your solutions of problems 4-7 (you do not need to run the code). Problems 1-3 also need to be solved on paper.

1. What is the output of the following algorithm? The initial value of T_ptr is the address of the root of the tree sketched below?

   ```cpp
   void T_traversal(TNode *T_ptr)
   {
       if(T_ptr!=NULL && T_ptr->data!=30)
       {
           cout << T_ptr->data << endl;
           T_traversal(T_ptr->left_link);
           cout << T_ptr->data << endl;
           T_traversal(T_ptr->right_link);
           cout << T_ptr->data << endl;
       }
   }
   ```

![Tree Diagram]

2. Provide an example of an input sequence, so that the tree, which is the result of running the insert function provided below is the full binary tree of depth 4.

   ```cpp
   TNode* insert(TNode* T_ptr, TNode* new_ptr)
   {
       if(T_ptr==NULL)
           return new_ptr;
       if(new_ptr->data < T_ptr->data)
           T_ptr->left_link=insert(T_ptr->left_link,new_ptr);
       else
           T_ptr->right_link=insert(T_ptr->right_link,new_ptr);
       return T_ptr;
   }
   ```
What should be the input sequence so that the resulting tree is the one of problem number 1?

3. Construct the Huffman tree for characters a,b,c,d,e and frequencies 5,10,20,25,40.

4. Write a function, which takes the address of the root of a tree as an input argument and computes the number of nodes with two children (inside this tree).

5. Write a function, which in a given binary tree computes the lengths of the shortest and the longest paths from the root to its leaves.

6. Write a function, which creates a full binary tree of depth d in just one tree traversal.
   Recall: In a full binary tree of depth d every node of depth smaller than d has two children and every node of depth d is a leaf.

7. Write a Boolean type function, which will check whether a given binary tree is a binary search tree. Assume that keys are integers. Assume that nodes have the following structure:
   ```
   struct TNode
   {
       Key key;
       Item data;
       TNode *left_link;
       TNode *right_link;
   };
   ```
   Recall: A binary search tree is a binary tree that has a key associated with each of its nodes, with the additional property that the key in any node is larger than the keys in all nodes of that node’s left sub-tree and smaller than the keys in all nodes of that node’s right sub-tree.