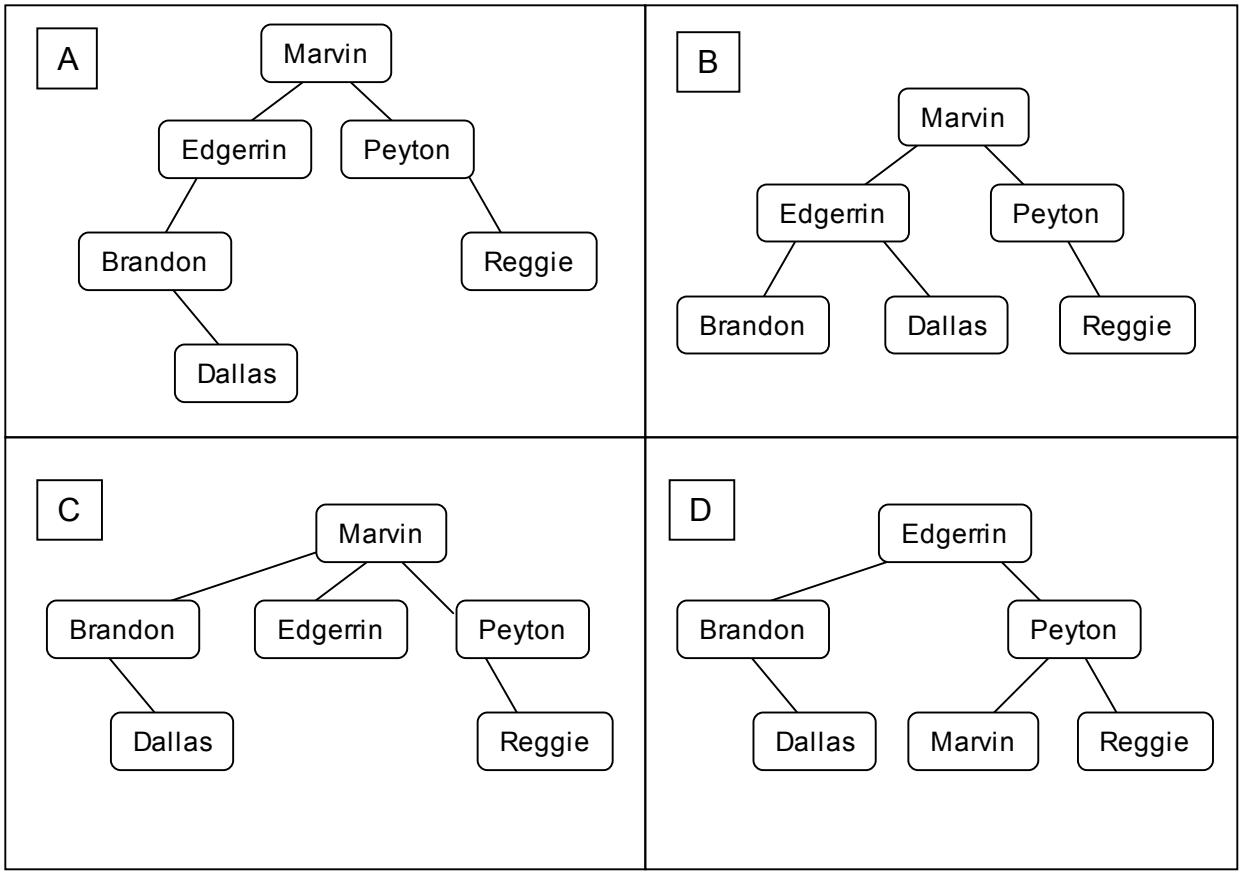
CS140 Final Exam - Fall, 2005

Jim Plank.

Answer All Questions.
Put your answers on a separate sheet of paper -- do not hand in this exam.
Put your name and email address on all sheets of paper that you hand in.

I will be grading this exam strictly.
Double-check your answers to make sure you've done things well.

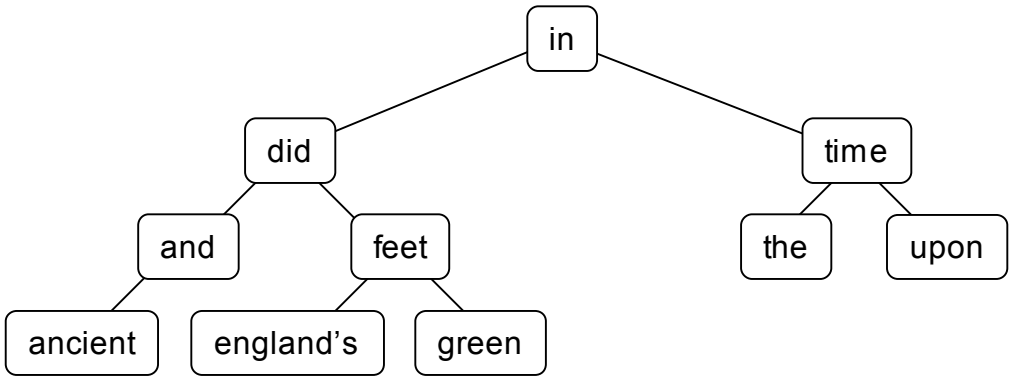
Question 1: (6 points) For each of the following trees, state whether or not the tree is a valid AVL tree. If it is not a valid AVL tree, state why it is not, and be as precise as possible in your answer (I.e. if a necessary condition is false, state *where* it is false, not just that it is false).



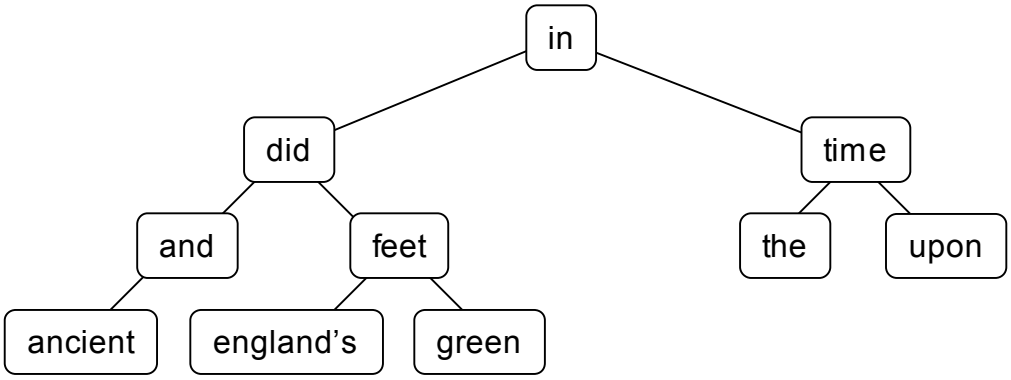
Question 2: (3 points) In tree "C" above, print out the names using a preorder tree traversal, and using a postorder tree traversal.

CS140 Final Exam - Fall, 2005 - Page 2

Question 3: (4 points) Assume that the tree below is an AVL tree. Show the tree that results when you insert the word "from" into the tree.



Question 4: (4 points) Assume that the tree below is a splay tree. Show the tree that results when you insert the word "hills" into the tree.



Question 5: (3 points) Explain the running time of insertions in splay trees in terms of big-O complexity.

Question 6: (4 points) What two criteria make for a good hash function?

CS140 Final Exam - Fall, 2005 - Page 3

```
main()
{
  IS is;
  JRB tmp, t, node;
  int i;

  t = make_jrb();
  is = new_inputstruct(NULL);

  while (get_line(is) >= 0) {
    tmp = t;
    for (i = 0; i < is->NF; i++) {
      node = jrb_find_str(tmp, is->fields[i]);
      if (node == NULL) {
        node = jrb_insert_str(tmp, strdup(is->fields[i]),
                              new_jval_v((void *) make_jrb()));
      }
      tmp = (JRB) node->val.v;
    }
  }
  print_tree(t, 0);
}
```

Question 7: (8 points) The above program, **mltree.c**, makes a big recursive red-black tree in the following manner: For each line of standard input, it inserts each word in the line into a different level of the red-black tree. For example, the first word goes into the top level of the tree. The `val` field of the node for the first word is a red-black tree into which the second word of the line goes. The `val` field of the node in that tree for the second word is a red-black tree for the third word. Etc.

Then, **mltree** performs a pre-order traversal of the tree, printing out each word in the tree on its own line, indented *X* spaces, where *X* is the level of the word. For example, suppose **input.txt** is the file on the left. Then the box on the right shows the output of **mltree** when **input.txt** is standard input:

input.txt

```
mai ai hee
mai ai huu
mai ai haa
mai ai haa haa

vrei sa pleci dar nu ma
nu ma iei
nu ma
nu ma iei
nu ma
nu ma
nu ma iei
```

mltree < input.txt

```
mai
 ai
  haa
   haa
  hee
  huu
nu
 ma
  iei
vrei
 sa
  pleci
   dar
    nu
     ma
```

Write the procedure **print_tree**().

CS140 Final Exam - Fall, 2005 - Page 4

Prototypes:

From fields.h

```
#define MAXLEN 1001
#define MAXFIELDS 1000

typedef struct inputstruct {
  char *name;              /* File name */
  int line;                /* Line number */
  char text1[MAXLEN];      /* The line */
  int NF;                  /* Number of fields */
  char *fields[MAXFIELDS]; /* Pointers to fields */
  …
} *IS;

extern IS new_inputstruct(char *filename/);
extern int get_line(IS is);
```

From jval.h

```
typedef union {
    int i;
    long l;
    float f;
    double d;
    void *v;
    char *s;
    …
  } Jval;

extern int jval_i(Jval);
extern long jval_l(Jval);
extern float jval_f(Jval);
extern double jval_d(Jval);
extern void *jval_v(Jval);
extern char *jval_s(Jval);
```

From jrb.h

```
typedef struct jrb_node {
  struct jrb_node *flink;
  struct jrb_node *blink;
  Jval key;
  Jval val;
  …
} *JRB;

#define jrb_traverse(ptr, lst) \
  for(ptr = jrb_first(lst); ptr != jrb_nil(lst); ptr = jrb_next(ptr))

extern JRB make_jrb();   /* Creates a new rb-tree */

/* Creates a node with key key and val val and inserts it into the tree.
   jrb_insert_str uses strcmp() as comparison funcion.  jrb_insert_int uses <>=,
   jrb_insert_gen uses func() */

extern JRB jrb_insert_str(JRB tree, char *key, Jval val);
extern JRB jrb_insert_int(JRB tree, int ikey, Jval val);
extern JRB jrb_insert_dbl(JRB tree, double dkey, Jval val);
extern JRB jrb_insert_gen(JRB tree, Jval key, Jval val, int (*func)(Jval,Jval));

/* returns an external node in t whose value is equal k. Returns NULL if
   there is no such node in the tree */

extern JRB jrb_find_str(JRB root, char *key);
extern JRB jrb_find_int(JRB root, int ikey);
extern JRB jrb_find_dbl(JRB root, double dkey);
extern JRB jrb_find_gen(JRB root, Jval, int (*func)(Jval, Jval));
```