Problem 37. (10 points):

Consider the following C declaration:

```
struct Node{
   char c;
   double value;
   struct Node* next;
   int flag;
   struct Node* left;
   struct Node* right;
};

typedef struct Node* pNode;

/* NodeTree is an array of N pointers to Node structs */
pNode NodeTree[N];
```

A. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a `Node` struct. Mark off and label the areas for each individual element (there are 6 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment).

Assume the Linux alignment rules discussed in Class 9. **Clearly indicate the right hand boundary of the data structure with a vertical line**.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                                                                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

B. For each of the four C references below, please indicate which assembly code section (labeled A – F) places the value of that C reference into register `%eax`. If no match is found, please write “NONE” next to the C reference.

The initial register-to-variable mapping for each assembly code section is:

```
  %eax = starting address of the NodeTree array
  %edx = i
```

```
-----------------------------------------------------------------
C References:

1.  ______ NodeTree[i]->flag

2.  ______ NodeTree[i]->left->left->c

3.  ______ NodeTree[i]->next->next->flag

4.  ______ NodeTree[i]->right->left->left
-----------------------------------------------------------------
```

```
Linux/IA32 Assembly:

A.      sall $2, %edx                   B.  sall $2,%edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl 16(%eax),%eax                  movl (%eax),%eax
                                            movl 24(%eax),%eax
                                            movl 20(%eax),%eax
                                            movl 20(%eax),%eax

C:      sall $2,%edx                    D:  sall $2,%edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl 20(%eax),%eax                  movl (%eax),%eax
        movl 20(%eax),%eax                  movl 16(%eax),%eax
        movsbl (%eax),%eax

E:      sall $2, %edx                   F:  sall $2, %edx
        leal (%eax,%edx),%eax               leal (%eax,%edx),%eax
        movl (%eax),%eax                    movl (%eax),%eax
        movl 16(%eax),%eax                  movl 12(%eax),%eax
        movl 16(%eax),%eax                  movl 12(%eax),%eax
        movl 20(%eax),%eax                  movl 16(%eax),%eax
```