The Activation Record Concept

Handout written by Jerry Cain.

Based on the following type declarations:

```
struct flashyauto {
   struct flashyauto *bmw;
   struct flashyauto **yugo
   short lexus[4];
   char audi[8];
};
```

Draw the activation record for the above `struct`, labelling field names, types, sizes and offsets from its base address. Using that information, draw out the stack frame for the following function call (ignoring return information that you may have read about but haven't seen in lecture yet) and then trace through the code and show how memory within (and perhaps outside of) the activation record is updated. In particular, make it clear what bytes are manipulated and updated by the last of the three statements.

```
void TestDrive()
{
   int numCarsSold;
   char *dealerName;
   struct flashyauto wheels[2];
   struct flashyauto *keys[2];

   dealerName[0] = 'A';
   keys[numCarsSold] = wheels + 1;
   *((short *) wheels[2].yugo) *= *((char *) wheels);
}
```

Parameters on the Stack

Trace through the following short, recursive program, and draw the state of the stack just before the deepest recursive call exits.

```
typedef struct node {
   int value;
   struct node *next;
} node;

main()
{
   node cells[3];
   int i, numCells = sizeof(cells)/sizeof(cells[0]);
   int sum;

   for (i = 0; i < numCells; i++) {
      cells[i].value = RandomInteger(50, 100);
      cells[i].next = &cells[i+i];
   }

   cells[numCells - 1].next = NULL;
   ReversePrint(cells);
}

static void ReversePrint(node *list)
{
   if (list == NULL) return;
   ReversePrint(list->next);
   printf("%d", list->value);
}
```