1) (10 pts) DSN (Dynamic Memory Management in C)

Consider the following struct, which contains a string and its length in one nice, neat package:

```
typedef struct smart_string {
  char *word;
  int length;
} smart_string;
```

Write a function that takes a string as its input, creates a new *smart_string* struct, and stores a **new copy of that string** in the *word* field of the struct and the length of that string in the *length* member of the struct. The function should then return a pointer to that new *smart_string* struct. Use dynamic memory management as necessary. The function signature is:

```
smart_string *create_smart_string(char *str) {

   smart_string *s = malloc(sizeof(smart_string));

   s->length = strlen(str);

   s->word = malloc(sizeof(char) * (s->length + 1));

   strcpy(s->word, str);

   return s;

}
```

Grading (7 pts): 1 pt for smart_string declaration, 1 pt for first malloc, 1 pt for setting length correctly, 1 pt for sizeof(char) in second malloc, 1 pt for (s->length + 1) in second malloc, 1 pt for using strcpy, and 1 pt for correct return statement. They can also use calloc(). Please deduct a point for other large, obvious errors (such as using the . operator instead of the -> operator).

Now write a function that takes a *smart_string* pointer (which might be NULL) as its only argument, frees all dynamically allocated memory associated with that struct, and returns NULL when it's finished.

```
smart_string *erase_smart_string(smart_string *s) {

   if (s != NULL)
   {
      free(s->word); // This is safe, even if word is NULL.
      free(s);
   }
   return NULL;
}
```

Grading (3 pts): 1 pt for checking s != NULL, 1 pt for free(s->word), and 1 pt for free(s).