1 Un commented Code? Yuck!

The following functions work correctly (note: this does not mean intelligently), but have no comments. Document the code to prevent it from causing further confusion.

1. /* Returns the sum of the first N elements in ARR. */
   int foo(int *arr, size_t n) {
       return n ? arr[0] + foo(arr + 1, n - 1) : 0;
   }

2. /* Returns the number of zeroes in the first n elements of arr. */
   int bar(int *arr, size_t n) {
       int sum = 0, i;
       for (i = n; i > 0; i--) {
           sum += !arr[i - 1];
       }
       return sum;
   }

3. /* Does nothing. */
   void baz(int x, int y) {
       x = x ^ y;
       y = x ^ y;
       x = x ^ y;
   }

2 Programming with Pointers

Implement the following functions so that they perform as described in the comments.

1. /* Swaps the value of two ints outside of this function. */
   void swap(int *x, int *y) {
       int temp = *x;
       *x = *y;
       *y = temp;
   }

2. /* Returns the number of characters in a string. Does not use strlen. */
   int mystrlen(char* str) {
       int count = 0;
       while(*str++){ //make sure to explain to students what’s happening here
           count++;
       }
       return count;
   }
3 Problem?

The following code segments may contain logic and syntax errors. Find and correct them.

1. /* Returns the sum of all the elements in SUMMANDS. */
   int sum(int* summands) { // int sum(int* summands, unsigned int n) {
   int sum = 0;
   for (int i = 0; i < sizeof(summands); i++) // for (int i = 0; i < n; i++)
   sum += *(summands + i);
   return sum;
   }

2. /* Increments all the letters in the string STRING, held in an array of length N.
   * Does not modify any other memory which has been previously allocated. */
   void increment(char* string, int n) { //more of a security concern
   for (int i = 0; i < n; i++) // for (i = 0; i<n && string[i] != 0; i++)
   *(string + i)++; // string[i]++;
   
   // consider the corner case of incrementing 0xFF
   }

3. /* Overwrites an inputted string with ‘‘61C is awesome!’’ if there’s room.
   * Does nothing if there is not. Assume that srcLength correctly represents
   * the length of src. */
   void CS61C(char* src, size_t srcLength) {
   char *srcptr, replaceptr; // char *srcptr, *replaceptr;
   char replacement[16] = ‘‘61C is awesome!’’;
   replaceptr = replacement;
   if (srcLength >= 16) {
   for (int i = 0; i < 16; i++)
   *src++ = *replacement++;
   }
   
   // “char *srcptr, replaceptr” initializes a char pointer and a char.
   // Not two char pointers.
   // “char *srcptr, replaceptr” is not the same as “char *srcptr, *replaceptr”.

   // “char *srcptr, replaceptr” initializes a char pointer and a char.
   // Not two char pointers.
   // “char *srcptr, replaceptr” is not the same as “char *srcptr, *replaceptr”.

   // “char *srcptr, replaceptr” initializes a char pointer and a char.
   // Not two char pointers.
   // “char *srcptr, replaceptr” is not the same as “char *srcptr, *replaceptr”.