```c
//
// Class Activity: Format String Vulnerabilities
//

//
// Draw the call stack contents (with stack frames for main() and printf() functions)
// when the call to printf() is currently active.
//
// Assume that local variables are allocated storage in the stack frame (high to low
// address) in the order of their declaration in the source code.
//
// Include va_list ptr in your diagram.
//

#include <stdio.h>

int main()
{
   int x = 100;
   int y = 25;
   char name1[10] = "John Doe";
   char *name2 = "Bob Smith";

   printf("x: %d, y: %d, Name1: %s, Name2: %s\n", x, y, name1, name2);

   return 0;
}

//
// Write a function named "totstrlen" that takes a variable number of strings as
// its arguments, and returns their total length.
//
// Here is a sample call to this function which returns 46
//
// totstrlen(3,"Grand Valley State University","Allendale","Michigan")
//
// Look at the example code (myprint.c) on pages 132-133 in the book
//

#include <stdio.h>
#include <stdarg.h>
#include <string.h>

int totstrlen(int narg, ...)
{











}

int main()
{
    int len = totstrlen(3,"Grand Valley State University","Allendale","Michigan");
    printf("Total length of the strings: %d\n",len);

    len = totstrlen(4,"Computer","security","is","so much fun!");
    printf("Total length of the strings: %d\n",len);
}
```