/\* homework 1 for embedded systems

Brian Cullinan

\*/

#include <stdio.h> /\* TEMPORARY FOR DEBUG printf, stderr, fprintf \*/

/\* 1. \*/

int fmt\_long(char\* s, long d)

{

int rtvalue = 0;

int significant\_zeros = 0;

long divider = 1000000000;

long temp = d;

if(d < 0)

{

if(s != NULL) s[0] = '-';

divider = -divider;

rtvalue++;

}

int i;

for(i = 10; i > 0; i--)

{

if(temp / divider != 0 || significant\_zeros == 1)

{

if(s != NULL) s[rtvalue] = '0' + temp / divider;

significant\_zeros = 1;

rtvalue++;

}

temp = d % divider;

divider = divider / 10;

}

if(s != NULL) s[rtvalue] = '\0';

return rtvalue;

}

/\* 2. \*/

int fmt\_double(char \*s, double d)

{

unsigned long long bits = \*(unsigned long long \*)&d;

int negative = (bits) >> 63;

int exponent = (bits << 1) >> 53;

unsigned long long mantissa = ((bits << 12) >> 12) | ((uint64\_t)1 << 52);

int rtvalue = fmt\_long(s, mantissa >> (52 - (exponent - 1023)));

return rtvalue;

}

int main(void)

{

char numbers[10];

fmt\_long(numbers, 21474836);

fmt\_double(numbers, 123456789.125);

printf("%s", numbers);

return 0;

}

/\* 3.

Because rounding error is difficult to handle.

\*/

/\* 4. \*/

