#  Preprocessor Homework

1. Write a preprocessor directive to accomplish each of the following:

a) Define symbolic constant TRUE to have the value 1.

#define TRUE 1

b) Define symbolic constant FALSE to have the value 0.

#define FALSE 0

c) If symbolic constant YES is defined, undefine it and redefine it as 1.

#if defined( YES ) #undef YES

#define YES 1

#endif

or

#ifdef YES #undef YES

#define YES 1

#endif

d) Define a macro, AREA, that takes a single argument and computes the area of a square.

#define AREA (x) ((x)\*(x))

e) Define macro SUM that will sum two numbers.

#define SUM (x,y) ((x)+(y))

f) Define macro MIN2 that will determine the smallest of two numeric values.

#define MIN2 (x,y) ((x)<(y)?(x):(y))

2. Write a macro that computes the remainder when n is divided by 4.

#define MOD(n) ((n)%4)

3. Let DOUBLE be the following macro:

#define DOUBLE(x) 2\*x

1. What is the value of DOUBLE(1+2)?

4

1. What is the value of 4/DOUBLE(2)?

4

1. Fix the definition of DOUBLE.

#define DOUBLE(x) (2\*(x))

4. For each of the following macros, give an example that illustrates a problem with the macro and show how to fix it.

a) #define AVG(x, y) (x+y)/2

One problem stems from the lack of parentheses around the replacement list. For example, the statement:

a = 1/AVG(b, c);

will be replaced by

a = 1/(b+c)/2;

Even if we add the missing parentheses, the macro still has problems, because it needs parentheses around x and y in the replacement list. The preprocessor will turn the statement:

a = AVG(b<c, c>d);

into

a = ((b<c+c>d)/2);

which is equivalent to

a = ((b<(c+c)>d)/2);

Here's the final (corrected) version of the macro:

#define AVG(x,y) (((x)+(y))/2)

b) #define AREA(x, y) (x) \* (y)

The problem is the lack of parentheses around the replacement list. For example,

a = 1/AREA(b, c);

becomes

a = 1/(b)\*(c);

Here's the corrected macro:

#define AREA(x,y) ((x)\*(y))