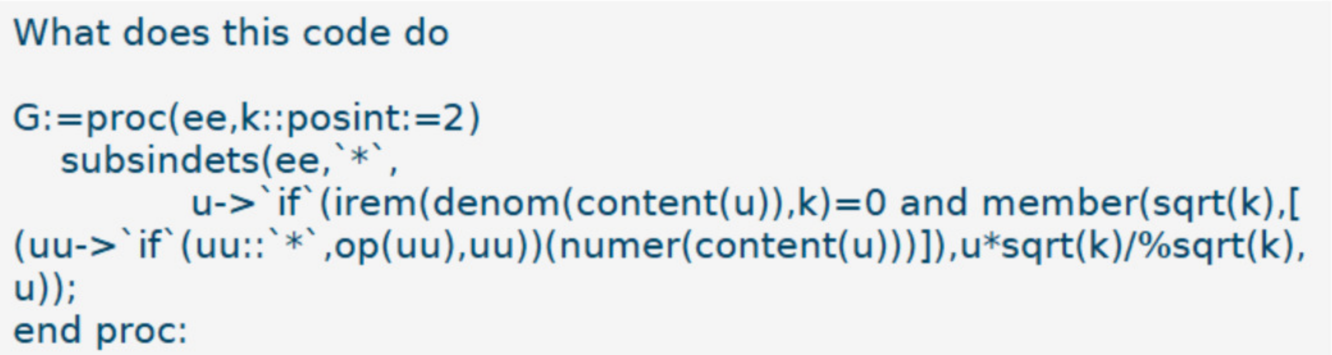

What does this code do

```
G:=proc(ee,k::posint:=2)
   subsindets(ee,`*`,
            u->`if`(irem(denom(content(u)),k)=0 and member(sqrt(k),[
(uu->`if`(uu::`*`,op(uu),uu))(numer(content(u)))]),u*sqrt(k)/%sqrt(k),
u));
end proc:
```

You defined a Maple procedure G that walks multiplicative subexpressions and conditionally multiplies certain factors by a quotient involving sqrt(k). Below I explain step by step what each part does, show a short example of its effect, point out a pitfall (the use of %), and offer a safer clearer version you can use instead.

Signature: G := proc(ee, k::posint := 2)

Line-by-line meaning:

- subsindets(ee, `*`, u -> ...): visit every subexpression of ee whose top-level operator is multiplication (operator ``*``) and call it u. Each such u is a candidate for replacement by the lambda-body result.
- content(u): returns the content of u. For rational expressions this isolates the rational factor (Maple's content normally used for polynomials; here it is used to extract the rational multiplier part of u).
- denom(content(u)) and numer(content(u)): the denominator and numerator of that content (as integers or integer factors when content is rational).
- irem(denom(content(u)),k)=0: tests whether the denominator of that content is divisible by k (i.e. remainder 0 on integer division by k).
- (uu -> if(uu::`*`, op(uu), uu))(numer(content(u))): if the numerator is itself a product, this produces the operand list of that product; otherwise it yields the numerator as a single object. The result is put into a one-element list constructor in the original code and then tested with member(sqrt(k), ...).
- member(sqrt(k), [...]): checks whether one of those factors equals sqrt(k).
- So the combined condition checks: “the denominator of the (rational) content of u is divisible by k AND the numerator of that content includes sqrt(k) as a factor.”
- If the condition is true the code returns u*sqrt(k)/%sqrt(k) as the replacement; otherwise it returns u unchanged.

Intent (interpretation): the procedure looks for multiplicative subexpressions that contain a rational factor whose denominator is divisible by k while the numerator contains a factor sqrt(k). For those factors it multiplies the subexpression by sqrt(k)/%sqrt(k). That multiplication has no algebraic effect if %sqrt(k) is the same object as sqrt(k), but because the code uses the special symbol % (the “last result” reference in Maple) the replacement can produce expressions that explicitly carry a % reference.

Using `sqrt(k)/%sqrt(k)` is fragile because % refers to the previous Maple output and is not a stable symbol; the code therefore depends on worksheet evaluation history and may produce odd results like `.../%sqrt(2)`. Likely the author intended to multiply by sqrt(k) /sqrt(k) (which is 1) but to force a change in internal representation so that the sqrt(k) factor becomes an explicit part of the product rather than hidden inside the content. A safer explicit approach should avoid %.

```
# Example showing how G behaves (from Maple run):
# expr := (3*sqrt(2)*x)/4;
# G(expr,2)  => returns something like 3/2*x/%sqrt(2) (shows the effect and the % usage)
# (The preceding was produced by Maple in a test run.)

```

Safer clearer alternative

If the goal is to force sqrt(k) to appear as an explicit multiplicative factor (without using %), introduce a local symbol S := sqrt(k) and use it in the replacement. For example:

```
G2 := proc(ee, k::posint := 2)
    local S;
    S := sqrt(k);
    subsindets(ee, `*`, u -> if irem(denom(content(u)), k) = 0 and member(S, [op(numer(content(u)))]) then u*S/S else u fi);
end proc:

# Example to run in Maple:
# expr := (3*sqrt(2)*x)/4:
# G2(expr,2);

```

Notes about the alternative:

- Using a local variable S bound to sqrt(k) avoids dependence on the worksheet's “last result” (`%`).
- Multiplying by S/S is algebraically 1, but it forces Maple to show sqrt(k) as an explicit factor in the product (which appears to be the original purpose).
- You may need to adapt the operand extraction if numer(content(u)) is not a product (the code above uses `op(numer(content(u)))` and relies on Maple returning a sequence of operands; wrap into a list if desired).

Summary

1. G traverses multiplicative subexpressions and, when it finds a rational content whose denominator is divisible by k and whose numerator includes sqrt(k), it multiplies that subexpression by sqrt(k)/%sqrt(k).
2. The use of % (the “last result” reference) is brittle and will produce expressions referencing %sqrt(k); this is probably unintended. Replace % with a stable local name (e.g. set S := sqrt(k) and use S/S) to get predictable, worksheet-history-independent behavior.