Equalities with Radicals

A solution to the problem of making radical + value = radical + value style problems procedurally that remain suitably "nice" for precalc students.

It is surprisingly difficult to random-generate a nice version of the classic $\sqrt{f(x)} = \sqrt{g(x)} + C$ style problems that maintain nice solveability by hand for a student at this level. Austin Jacobs and Jason Nowell coded this version that abuses the fact that we can force two parabola to intersect each other on an integer lattice in predictable places, then generate the radicals by using inverse functions of the parabolas while inferring the solutions by applying the inverse process to the integer lattice geometrically.

Apologies that the code is rather opaque - it requires a picture that no longer exists.

Below is the explanation for the code as per Austin,

Explanation. *The core idea here is that square roots with a linear term as their argument can be thought of as the "inverse" of a parabola. So what we are going to do is take two parabolas and then just draw a horizontal line at the y value we want to be our solution. The first parabola will be* $F(x) = a1(x - b1)^2 + c1$. *As long as we choose a1, b1, and c1 to be integers, we know that putting in integers for x will force F(x) to be an integer (eg if* $x = b1 + 2$ *then* $F(x) = a1 * 4 + c1$*). So we choose three integers and then pick a value for x-b1, which we'll call p1w. This will give us the point* $(b1 + p1w, p1h)$ *with a guarantee of both coordinates being integers.*

Now we draw our horizontal line $y = p1h$ *which we know goes through that point (indeed, this will end up being our solution). We pick a horizontal distance to travel along this path, say C, and we specifically choose it to be an integer. This will be our "+C" in the initial problem statement.*

We now have another integer point $(b1 + p1w + C, p1h)$. *This is our jumping point for the second parabola, G(x). The only thing we need to do now is pick a2, b2, c2, and d2, however we don't need to pick all of these. Once we make some choices the rest will be fixed. For simplicity, choose a2 and d2. This will let us know that our second parabola will have its vertext at* $(b1 + p1w + C - d2, p1h - a2 * d2^2)$ *and we only need to sort out from here what the appropriate b2 and c2 need to be. Since we have the vertex of the parabola, we know that b2 is the x value of that point and c2 is just the y value.*

Finally, we get the display expressions by taking the function inverses of F and G.

Learning outcomes:

The Finished Product

Problem 1 *What is the **sum** of the answers to the following equality?*

$$\sqrt{\frac{1}{5}x - 1} + 8 = \sqrt{x + 15} + 4$$

The sum of solutions is: $\boxed{10}$.

Feedback(*`attempt`*): *Recall that you need to isolate one of the radicals, then square both sides. Once you expand everything out, you will need to re-isolate the remaining radical and square both sides again, which should leave you with a quadratic. Once you solve the quadratic, you then need to check your solutions to see if they are extraneous or valid.*

The magic

Here is the sage code that generates the problem... as mentioned it is currently rather opaque.

```
\begin{sagesilent}

def RandInt(a,b):
    """ Returns a random integer in [‘a‘,‘b‘]. Note that ‘a‘ and ‘b‘ should be integers then
    """
    return QQ(randint(int(a),int(b)))
    # return choice(range(a,b+1))

def NonZeroInt(b,c, avoid = [0]):
    """ Returns a random integer in [‘b‘,‘c‘] which is not in ‘av‘.
        If ‘av‘ is not specified, defaults to a non-zero integer.
    """
    while True:
        a = RandInt(b,c)
        if a not in avoid:
            return a
```

```
p1ans1 = 100
p1ans2 = 200

while p1ans1>50 or p1ans2>50:
    # Make p(x)
    p1c1 = NonZeroInt(1,5)# a
    p1c2 = RandInt(1,5)# b
    p1c3 = RandInt(1,5)# c

    p1px = p1c1*(x-p1c2)^2 + p1c3

    p1w = RandInt(1,3)

    p1h = p1px(x=(p1c2+p1w))

    p1wprime = RandInt(2,5)

    p1bprime = RandInt(p1c2+1, 10)
    p1aprime = NonZeroInt(1,5,[p1c1])
    p1cprime = p1h - p1aprime*p1wprime^2

    p1gap = p1bprime + p1wprime -p1c2 - p1w#RandInt(p1bprime - p1c2 - p1w + 1, 2*(p1bprime -


    p1left = sqrt((x - p1c3)/p1c1) + p1c2 + p1gap
    p1right = sqrt((x - p1cprime)/p1aprime) + p1bprime


    p1d = p1c2+p1gap-p1bprime
    p1e = p1c3*p1aprime-p1cprime*p1c1-p1d^2*p1c1*p1aprime

    p1A = (p1c1-p1aprime)^2
    p1B = 2*(p1c1-p1aprime)*p1e - 4*p1d^2*p1aprime^2*p1c1
    p1C = p1e^2 + 4*p1d^2*p1aprime^2*p1c1*p1c3

    p1ans1 = (-p1B + sqrt(p1B^2 - 4*p1A*p1C))/(2*p1A)
    p1ans2 = (-p1B - sqrt(p1B^2 - 4*p1A*p1C))/(2*p1A)

    if p1left(x=p1ans1) == p1right(x=p1ans1):
        if p1left(x=p1ans2)==p1right(x=p1ans2):
            p1ans = p1ans1+p1ans2
        else:
            p1ans=p1ans1
    else:
        p1ans=p1ans2
```

```
    if p1ans1==p1ans2:
        if p1left(x=p1ans1)==p1right(x=p1ans1):
            p1ans=p1ans1
        else:
            p1ans=LatexExpr(r"DNE")


\end{sagesilent}
```