```
SkipMap rAdd (recursive add) starts at an entry in the top list and adds entries
with the key and value to the bottom list and also to extra ones determined by coin
flips.  If it adds to a list it returns the new entry.  Otherwise, it returns null.

I will name the methods rAdd1, rAdd2, rAdd3, rAdd4.  rAdd1 is the first call.  It
calls rAdd2 recursively and so forth.  They are all actually named rAdd, but the
numbering will help you understand which call we are talking about (I hope).

        B                          O              T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0


        B                             O              T

        B     F  G                    O              T  U

        B  C  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

```
        B                           O             *T

        B     F  G                  O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

rAdd1(*, N, 6) is called by put. put calls with top, and any entry in the topmost
list might be the top.

```
       *B                           O               T

        B     F  G                  O               T  U

        B  C  F  G        K     M  O     Q          T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0

rAdd1(*, N, 6) first calls find(*, N) on the current list and gets B.
```

```
        B                          O               T

       *B     F  G                 O               T  U

        B  C  F  G        K     M  O     Q         T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

rAdd1(*, N, 6) tests that this is a "skip Entry" (isSkipEntry() is true) and then
calls rAdd2(*, N, 6) recursively on the value (getEntry) of B.

```
        B                          O              T

        B     F  G               *O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0

rAdd2(*, N, 6) (the recursive call) calls find(*, N) and gets O.
```

```
        B                          O              T

        B     F  G                 O              T  U

        B  C  F  G        K     M *O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

rAdd2(*, N, 6) tests that this is a skip Entry then calls rAdd3(*, N, 6) on the
value of O.

```
        B                          O              T

        B     F  G                 O              T  U

        B  C  F  G        K    *M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0

rAdd3(*, N, 6) calls find(*, N) and gets M on this level.
```

```
        B                          O              T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L *M  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0
```

rAdd3(*, N, 6) tests that this is a skip Entry then calls rAdd4(*, N, 6) on the value of M.

```
        B                          O              T

        B     F  G                 O              T  U

        B  C  F  G        K     M  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M *O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  8  4  6  3  5  6  3  4  2  0

rAdd4(*, N, 6) calls find(*, N, 6) and gets O on this level.
```

```
           B                                   O                   T

           B       F   G                       O                   T   U

           B   C   F   G           K       M   O       Q           T   U   W       Z

       A   B   C   F   G   I   J   K   L   M  *O   P   Q   R   S   T   U   W   X   Z
       4   3   1   8   6   7   0   2   9   0   8   4   6   3   5   6   3   4   2   0
```

rAdd4(*, N, 6) tests that this entry is not a skip entry (isSkipEntry() is false) so
it is on the bottom. It can’t have key N because put won’t call rAdd if N is
already there.

```
         B                               O               T

         B     F  G                      O               T  U

         B  C  F  G        K     M       O     Q         T  U  W     Z

      A  B  C  F  G  I  J  K  L  M +N *O  P  Q  R  S  T  U  W  X  Z
      4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

rAdd4(*, N, 6) calls add(*, new Entry(N, 6)) and returns the result, which is the
new Entry with N (+).

```
        B                             O              T

        B     F  G                    O              T  U

        B  C  F  G        K    *M  N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M +N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

rAdd3 gets the new Entry (+) from rAdd4. Remember, it was pointing at M on its level.

Since it got back a new Entry, it flips a coin to decide if it will add a new Entry on its level. It gets heads (head() is true) so it calls add(*, new Entry(N, +)) and returns the result.

```
        B                             O              T

        B     F  G                   *O              T  U

        B  C  F  G        K     M +N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

rAdd2 gets back the new Entry. It is pointing at O on its level.

Since it got back a new Entry, it flips a coin to decide if it will add a new Entry
on its level. It gets TAILS (heads() is false) so it does not add a new Entry and
just returns null.

```
       *B                            O              T

        B     F  G                   O              T  U

        B  C  F  G        K     M  N  O     Q        T  U  W     Z

     A  B  C  F  G  I  J  K  L  M  N  O  P  Q  R  S  T  U  W  X  Z
     4  3  1  8  6  7  0  2  9  0  6  8  4  6  3  5  6  3  4  2  0
```

rAdd1 gets back null. It is pointing at B on its level.

Since it got back null, it does not add an entry on its level and returns null.