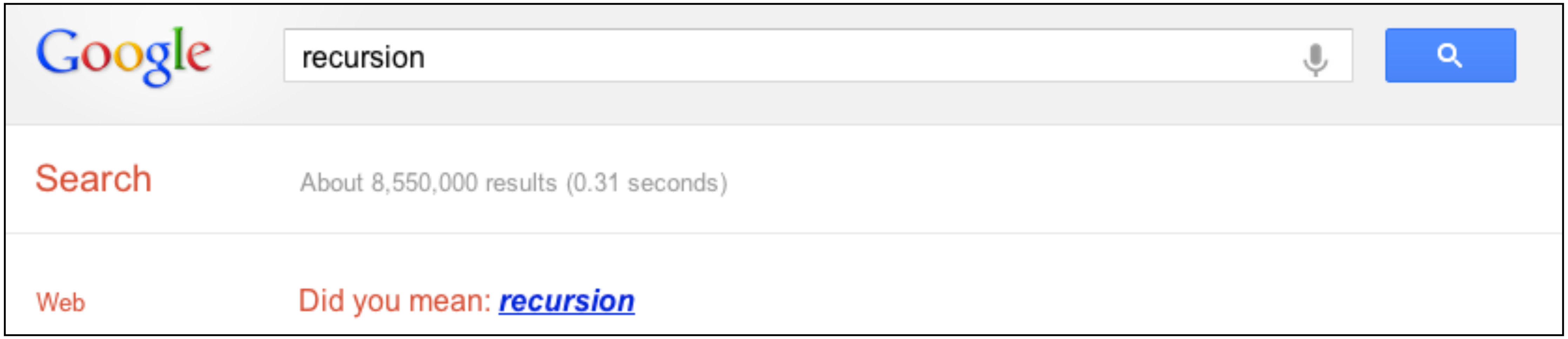
Recursion



recursion

Search

About 8,550,000 results (0.31 seconds)

Web

Did you mean: ***recursion***

(Why yes, I suppose I did.)

Last couple of days: *abstract.*

Today: code with code sauce.

A method defined...

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

A method defined...

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

Wait a minute...

A method defined...

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

$$N! =$$

...in terms of itself.

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

$$N! = N \cdot \underset{\text{Not so new after all.}}{\underline{(N-1)!}}$$

...in terms of itself.

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

$$1! = 1$$

$$N! = N \cdot \underset{\text{Not so new after all.}}{\underline{(N-1)!}}$$

...in terms of itself.

```
long secret1(long i) {
  if (i == 1) {
    return 1;
  }

  long c = secret1(i-1);
  return i * c;
}
```

$$1! = 1$$

$$N! = N \cdot (N - 1)!$$

Another one

```
long secret2(long i, long j) {
  if (j == 0) {
    return 1;
  }
  return i * secret2(i, j-1);
}
```

Another one

```
long secret2(int i, long j) {
  if (j == 0) {
    return 1;
  }
  return i * secret2(i, j-1);
}
```

$$m^0 = 1$$

$$m^n = m \cdot \left(m^{n-1}\right)$$

Another one

```
long secret2(int i, long j) {
  if (j == 0) {
    return 1;
  }
  return i * secret2(i, j-1);
}
```

$$m^0 = 1$$

$$m^n = m \cdot \left(m^{n-1}\right)$$

Yet Another one

```
long secret3(int i, int[] values) {
  if (i == values.length) {
    return 0;
  }
  return values[i] + secret3(i+1, values);
}
```

A pattern emerges

```
long secret1(long i) {
    if (i == 1) {
        return 1;
    }

    long c = secret1(i-1);
    return i * c;
}
```

```
long secret2(long i, long j) {
    if (j == 0) {
        return 1;
    }
    return i * secret2(i, j-1);
}
```

```
long secret3(int i, int[] values) {
    if (i == values.length) {
        return 0;
    }
    return values[i] + secret3(i+1, values);
}
```

A pattern emerges

```
long secret1(long i) {
    if (i == 1) {
        return 1;
    }

    long c = secret1(i-1);
    return i * c;
}
```

```
long secret2(long i, long j) {
    if (j == 0) {
        return 1;
    }
    return i * secret2(i, j-1);
}
```

```
long secret3(int i, int[] values) {
    if (i == values.length) {
        return 0;
    }
    return values[i] + secret3(i+1, values);
}
```

```
if some stopping condition
  return a value
store the result of a recursive call
compute the answer using that value
return the answer
```

Terminology

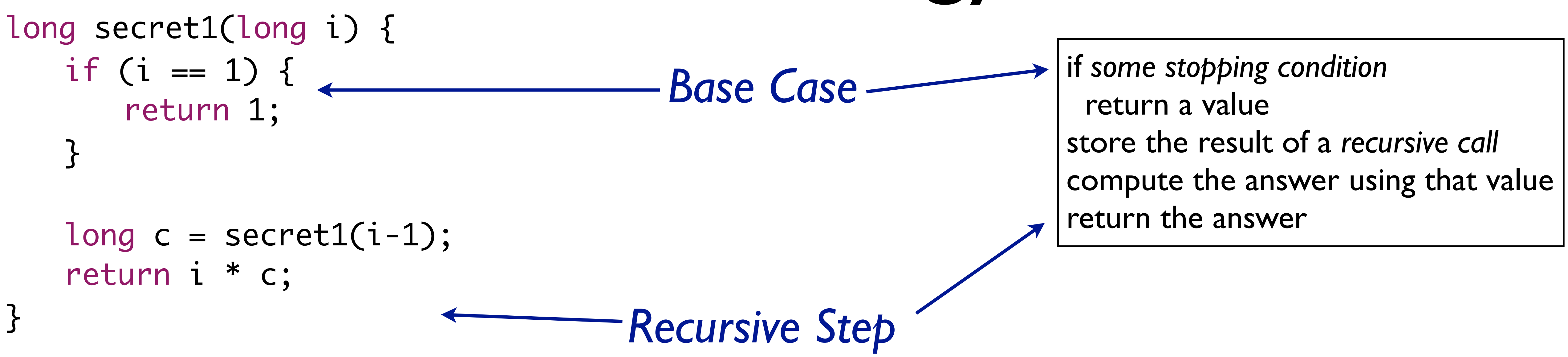
```
long secret1(long i) {
    if (i == 1) {
        return 1;
    }

    long c = secret1(i-1);
    return i * c;
}
```

Base Case

Recursive Step

if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Terminology



```
long secret1(long i) {
    if (i == 1) {
        return 1;
    }

    long c = secret1(i-1);
    return i * c;
}
```

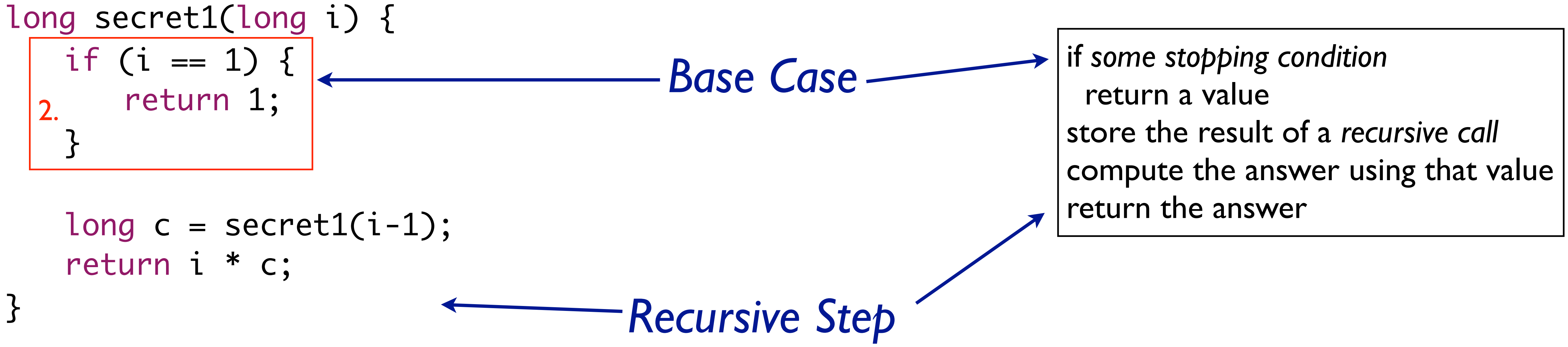
if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Mac's Patented Human Algorithm for Writing Recursive Algorithms

1. Figure out how your problem gets smaller

An integer gets smaller
or
You move one step further through an array
or
You move one step along a list.

Terminology



```
long secret1(long i) {
    if (i == 1) {
2.      return 1;
    }

    long c = secret1(i-1);
    return i * c;
}
```

Base Case

Recursive Step

if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Mac's Patented Human Algorithm for Writing Recursive Algorithms

1. Figure out how your problem gets smaller

An integer gets smaller
or
You move one step further through an array
or
You move one step along a list.

2. What's the smallest that can get?

Often 0, or 1, or an empty list.

Terminology

```
long secret1(long i) {
    if (i == 1) {
2.      return 1;
3.  }

    long c = secret1(i-1);
    return i * c;
}
```

Base Case

Recursive Step

if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Mac's Patented Human Algorithm for Writing Recursive Algorithms

1. Figure out how your problem gets smaller

An integer gets smaller
or
You move one step further through an array
or
You move one step along a list.

2. What's the smallest that can get?

Often 0, or 1, or an empty list.

3. That's your base case. *Write it!*

We grade on this. Also, demo coming up!

Terminology

```
long secret1(long i) {
   if (i == 1) {
2.     return 1;
3. }

   long c = secret1(i-1);    4.
   return i * c;
}
```

Base Case

Recursive Step

if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Mac's Patented Human Algorithm for Writing Recursive Algorithms

1. Figure out how your problem gets smaller

An integer gets smaller
or
You move one step further through an array
or
You move one step along a list.

2. What's the smallest that can get?

Often 0, or 1, or an empty list.

3. That's your base case. *Write it!*

We grade on this. Also, demo coming up!

4. Compute the answer to the one-smaller problem.

Recurse!

Terminology

```
long secret1(long i) {
   if (i == 1) {
2.     return 1;
3. }

   long c = secret1(i-1);    4.
5. return i * c;
}
```

Base Case

Recursive Step

if *some stopping condition*
  return a value
store the result of a *recursive call*
compute the answer using that value
return the answer

Mac's Patented Human Algorithm for Writing Recursive Algorithms

1. Figure out how your problem gets smaller

An integer gets smaller
or
You move one step further through an array
or
You move one step along a list.

2. What's the smallest that can get?

Often 0, or 1, or an empty list.

3. That's your base case. *Write it!*

We grade on this. Also, demo coming up!

4. Compute the answer to the one-smaller problem.

Recurse!

5. Compute the answer to the this-sized problem.

Demo time!

countAs

isPalindrome

http://codingbat.com/java/Recursion-1