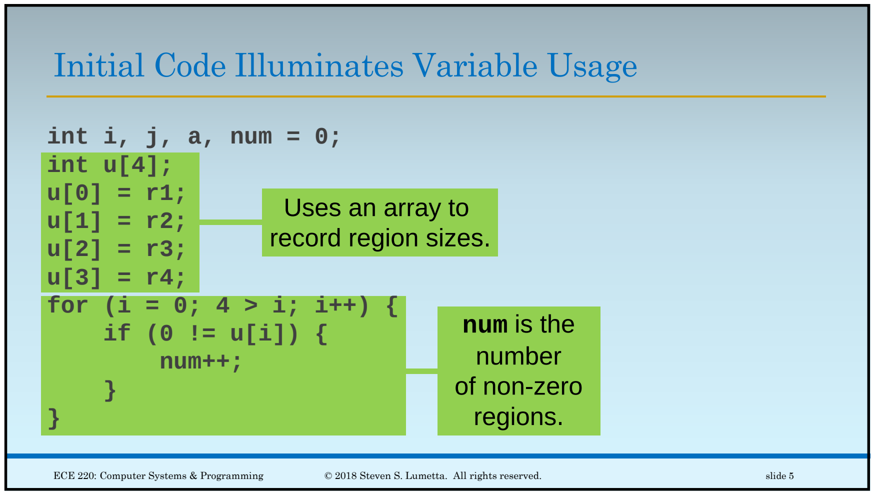
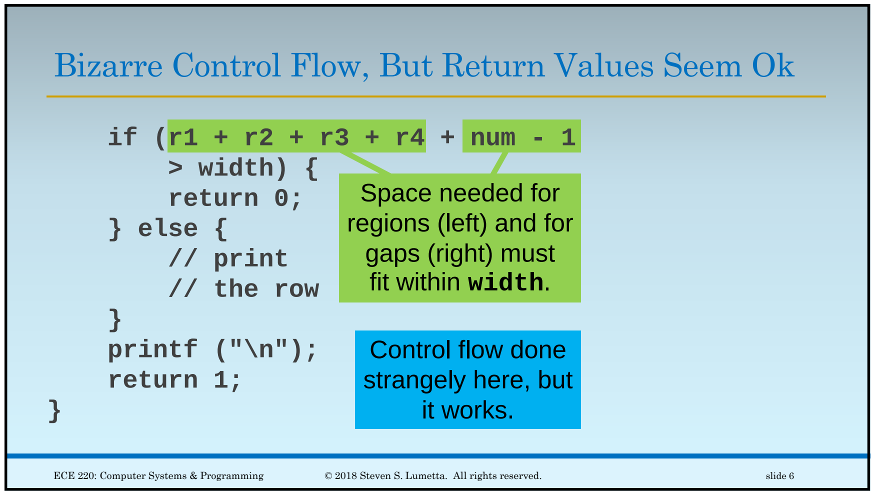


Initial Code Illuminates Variable Usage

```
int i, j, a, num = 0;
int u[4];
u[0] = r1;
u[1] = r2;
u[2] = r3;
u[3] = r4;
for (i = 0; 4 > i; i++) {
    if (0 != u[i]) {
        num++;
    }
}
```

Uses an array to record region sizes.

num is the number of non-zero regions.

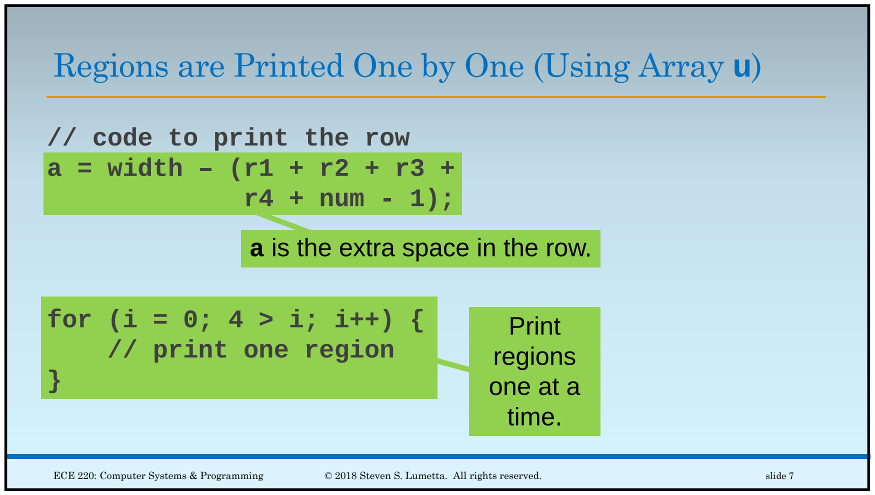


Bizarre Control Flow, But Return Values Seem Ok

```
    if (r1 + r2 + r3 + r4 + num - 1
        > width) {
        return 0;
    } else {
        // print
        // the row
    }
    printf ("\n");
    return 1;
}
```

Space needed for regions (left) and for gaps (right) must fit within **width**.

Control flow done strangely here, but it works.



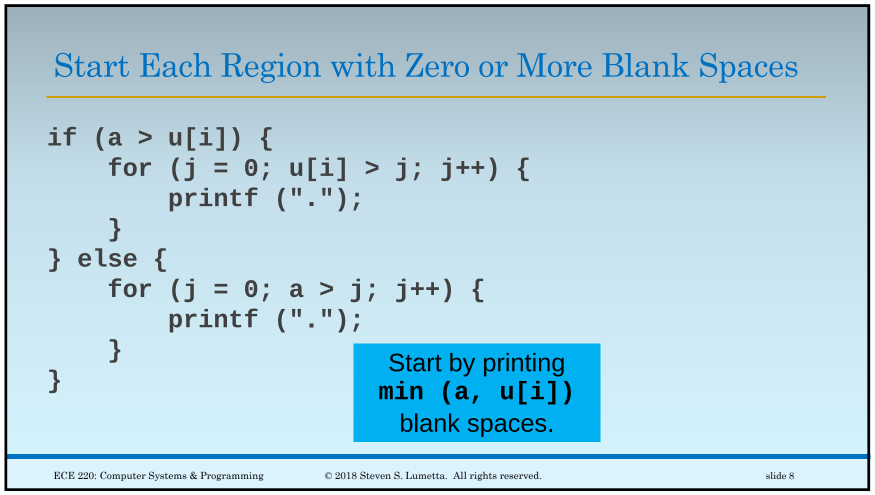
Regions are Printed One by One (Using Array **u**)

```
// code to print the row
a = width – (r1 + r2 + r3 +
             r4 + num - 1);
```

a is the extra space in the row.

```
for (i = 0; 4 > i; i++) {
    // print one region
}
```

Print regions one at a time.



Start Each Region with Zero or More Blank Spaces

```
if (a > u[i]) {
    for (j = 0; u[i] > j; j++) {
        printf (".");
    }
} else {
    for (j = 0; a > j; j++) {
        printf (".");
    }
}
```

Start by printing **min (a, u[i])** blank spaces.