Here's the output:

Total cookies eaten: 255
First three eaters ate 7 cookies.
Last four eaters ate 240 cookies.

Program Notes

First, notice the for loop in the sum_array() function:

```c
for (pt = begin; pt != end; pt++)
    total = total + *pt;
```

It sets `pt` to point to the first element to be processed (the one pointed to by `begin`) and adds `*pt` (the value of the element) to `total`. Then the loop updates `pt` to point to the next element. The process continues as long as `pt != end`. When `pt` finally equals `end`, it's pointing to the location following the last element of the range, so the loop halts.

Second, notice how the different function calls specify different ranges within the array:

```c
int sum = sum_arr(cookies, cookies + ArSize);
...
sum = sum_arr(cookies, cookies + 3); // first 3 elements
...
sum = sum_arr(cookies + 4, cookies + 8); // last 4 elements
```

The pointer value `cookies + ArSize` points the location following the last element. (The array has `ArSize` elements, so `cookies[ArSize - 1]` is the last element, and its address is `cookies + ArSize - 1`.) So the range `cookies, cookies + ArSize` specifies the entire array. Similarly, `cookies, cookies + 3` specifies the first three elements, and so on.

Note, by the way, that the rules for pointer subraction imply that, in `sum_arr()`, the expression `end - begin` is an integer value equal to the number of elements in the range.