```
count++; /* last execution of for */
count++; /* for return */ return tempsum;
}
```

Examining the program, we can see that if count's initial value is 0, it's final value will be 2n+3. Thus, each invocation of sum executes a total of 2n+3 steps. □

```
float sum(float list[], int n)
{
  float tempsum=0;
  int i;
  for(i=0; i<n; i++)
    count+=2;
  count+=3;
  return 0;
}
```

→ Simplified version

Ex 1.10 [Recursive summing of a list of numbers]: Find the step count

```
float rsum(float list[], int n)
{
  count++; /* for if conditional */
  if(n){
    count++; /* for return and rsum invocation */
    return rsum(list,n-1)+list[n-1];
  }
  count++;
  return list[0];
}
```

. To determine the step count for this function, we first need to figure out the step count for the boundary condition of n=0.

. We can see that when n=0 only the if conditional and the second return statement are executed. So, the total step count for n=0 is 2.