Example:

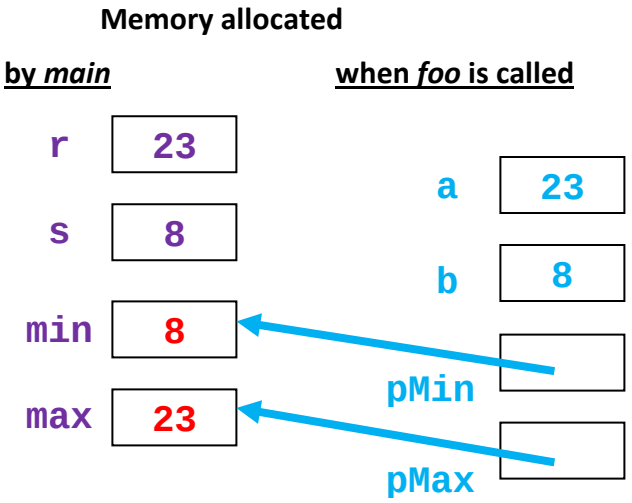
```
void foo(int a, int b, int* pMin, int* pMax) {
    if (a < b) {
        *pMin = a;
        *pMax = b;
    } else {
        *pMin = b;
        *pMax = a;
    }
}

void main() {
    int r = 23;
    int s = 8;

    int min;
    int max;

    foo(r, s, &min, &max);
}
```

```
// Function foo puts the smaller of a and b
// into pMin's pointee.  It puts the larger
// of a and b into pMax's pointee.
```



The **purple** happens because of the four declarations in ***main***: declaring variables allocates space (i.e., creates boxes). At this point, the boxes for the variables ***min*** and ***max*** (in ***main***) are still empty (which means that those variables contain garbage).

The **blue** happens when ***foo*** is *called*.

- Space is allocated for the parameters: ***a***, ***b***, ***pMin*** and ***pMax***.
- Those parameters are assigned values by copying the bits of the actual arguments (***r***, ***s***, ***&min*** and ***&max***) into the parameters.

The **red** happens when the *body of* ***foo*** *runs*.

- That body sets ***pMin***'s ***pointee*** to the smaller of ***a*** and ***b*** (here, that is 8) and ***pMax's pointee*** to the larger of ***a*** and ***b*** (here, that is 23).
- Since those pointees are the same boxes that the variables ***min*** and ***max*** in ***main*** refer to, respectively, the ***min*** and ***max*** variables in ***main*** are changed by the call to function ***foo***. Thus, this example illustrates how to send information back from a function to the caller by using pointers.

Finally, just to illustrate pointer assignment, suppose that ***foo*** had (for no good reason) the statement `pMin = pMax`. That would cause the *arrow* from ***pMin*** to point to the same place that the arrow from ***pMax*** points to, namely, to ***max***'s box.

If you have questions about the above example, please ask now!