Given the following C type definitions:

```c
struct l {
    char c1;
    char c2;
    int i;
    short s;
    double d[2];
};

struct r {
    short s;
    char c1;
    int i;
    double d[2];
    char c2;
};

void fillmem(char *p, int size)
{
    int i;

    for (i=0; i < size; i++)
        p[i] = i;
}

int fillStruct()
{
    struct l left;
    struct r right;

    fillmem((char *) &left, sizeof(struct l));
    fillmem((char *) &right, sizeof(struct r));
    /* here */
}
```
Fill in the blanks based on the state of `left` and `right` after the calls to `fillmem`. (Hint: you can use hexadecimal when convenient.)

a. What is the value of `left.s` on a little endian linux machine?

b. What is the value of `left.i` on a big endian linux machine?

c. What is the value of `right.c2` on a little endian windows machine?

d. What is the size of `left` on windows?

e. What is the size of `right` on linux?

f. How many wasted bytes are there in `left` on linux?

g. How many wasted bytes are there in `right` on windows?

h. How many bytes are needed for the smallest struct that contains the same fields as `struct left` on linux?