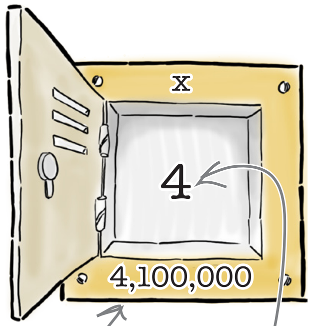
Using memory pointers

There are **three** things you need to know in order to use pointers to read and write data.

1 Get the address of a variable.
You've already seen that you can find where a variable is stored in memory using the **&** operator:

The %p format will print out the location in hex (base 16) format.

```
int x = 4;
printf("x lives at %p\n", &x);
```



& will find the address of the variable: 4,100,000.

But once you've got the address of a variable, you may want to store it somewhere. To do that, you will need a **pointer variable**. A pointer variable is just a variable that stores a memory address. When you declare a pointer variable, you need to say what kind of data is stored at the address it will point to:

This is a pointer variable for an address that stores an int.

```
int *address_of_x = &x;
```

2 Read the contents of an address.
When you have a memory address, you will want to read the data that's stored there. You do that with the ***** operator:

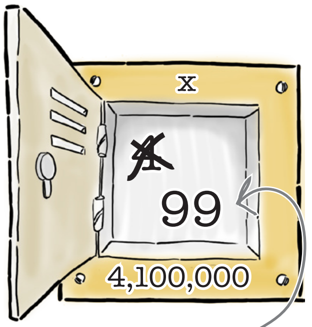
```
int value_stored = *address_of_x;
```

This will read the contents at the memory address given by address_of_x. This will be set to 4: the value originally stored in the x variable.

The * and & operators are opposites. The & operator takes a piece of data and tells you where it's stored. The * operator takes an address and tells you what's stored there. Because pointers are sometimes called *references*, the * operator is said to **dereference** a pointer.

3 Change the contents of an address.
If you have a pointer variable and you want to change the data at the address where the variable's pointing, you can just use the * operator again. But this time you need to use it on the **left side** of an assignment:

```
*address_of_x = 99;
```



This will change the contents of the original x variable to 99.

OK, now that you know how to read and write the contents of a memory location, it's time for you to fix the go_south_east() function.