```c
#include <stdio.h>

int sum(int a, int b){
    return a + b;
}

void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}

void checkCloning(){
    int a;
    printf("Initial value of this local variable is %d\n", a);
    a = 42;
    printf("Now the value of the local variable is %d\n", a);
}

void dummy(){
    float b = 63;
}

int max(int a, int b){
    return a > b ? a : b;
}

int min(int a, int b){
    // Functions can call each other :)
    // The following identity always holds
    // max(a,b) + min(a,b) = a + b
    return a + b - max(a,b);
}

// This function expects an address as input
// and returns the value stored at that address + 1
int findValue(int *a){
    printf("Value at address is %d\n", *a);
    return *a + 1;
}

int main(){
    int a[10] = {1,2,3,4,5,6,7,8,9,0};
    // Values returned by functions can be used anywhere
    // Be careful of return type though
    // sum function returns an integer so I should be able
    // to use its returned value to index an array :)
    printf("%d\n", a[sum(5,2)]);
    // or even in a math expression
    printf("%d\n", 17 % sum(5,2));

    // Values sent into a function as inputs
    // are copied. If the function changes those values
    // The original values are unaffected
    int p = 5, q = 2;
    printf("p = %d, q = %d\n", p, q);
    swap(p, q);
    printf("p = %d, q = %d\n", p, q);

    // Mr C clones himself everytime a function is called
    // even if the same function is called twice
    // Do not expect values you set inside a function to stay
    // the same when you call that function again
    // (unless the variable is a static variable)
    checkCloning();
    dummy();
    checkCloning();

    printf("Finding minimum using maximum %d\n", min(-5, -8));
```

```
    // Can pass an address to a function too
    // The input type should be a pointer in that case
    printf("Value returned is %d\n", findValue(&a[6]));
    return 0;
}
```