typedef struct {
    unsigned side;
} square;

void square_square(square *this, unsigned side) {
    this->side = side;
}

unsigned square_area(square *this) {
    return this->side * this->side;
}

unsigned things_and_stuff() {
    square s;
    square_square(&s, 32);
    unsigned area = square_area(&s);
    // destructor call
    return area;
}

things_and_stuff:
    push {lr}
    sub sp, sp, 4    // Making space for s
    // s is on offsets 0-3 of the stack
    mov a1, sp      // &s
class animal {
public:
    animal (unsigned age) : age_(age) {
        // remember to set up vtable pointer
    }

    virtual void speak() = 0;

    virtual unsigned age() {
        return age_;}

protected:
    unsigned age_;};
class fox : public animal {
public:
    fox(unsigned age) : animal(age) {
        // remember to set up vtable pointer
    }

    virtual void speak() {
        puts("Ring-ding-ding-ding-dingeringeding!");
    }
};
unsigned speak_and_get_age(animal *a) {
    a->speak();
    return a->age();
}
unsigned get_min_age() {
    fox f(16);
    unsigned age = speak_and_get_age(&f);
    return age / 2 + 7;
}
class animal {
public:
    animal (unsigned age) : age_(age) {
        // remember to set up vtable pointer
    }

    virtual void speak() = 0;

    virtual unsigned age() {
        return age_;  
    }

protected:
    unsigned age_;  
};

.data
animal_vtable:
    // pure virtual methods get a NULL (0) entry

.text
animal_vtable_ptr: .word animal_vtable

animal_animal:

animal_age:
class fox : public animal {
public:
    fox(unsigned age) : animal(age) {
        // remember to set up vtable pointer
    }

    virtual void speak() {
        puts("Ring-ding-ding-ding-dingereding!");
    }
};

.data
what_the_fox_say:
    .asciiz "Ring-ding-ding-ding-dingereding!"

fox_vtable:

.text
fox_vtable_ptr: .word fox_vtable

fox_fox:
    push {v1, lr}
    mov v1, a1
    bl animal_animal
    ldr a1, _____________
    str a1, [v1, ____ ]
    pop {v1, lr}
    bx lr

fox_speak:
unsigned speak_and_get_age(animal *a) {
    a->speak();
    return a->age();
}

speak_and_get_age:
unsigned get_min_age() {
    fox f(16);
    unsigned age = speak_and_get_age(&f);
    return age / 2 + 7;
}

get_min_age:
    push {lr}

    sub sp, sp, #8
    // f is on offsets 0-7 of the stack

    sdiv a1, a1, #2
    add a1, a1, #7

    add sp, sp, #8
    pop {lr}
    bx lr