```c
#include "environment.h"
#include "interpreter.h"
#include "C.tab.h"

extern VALUE* make_value_int(int);

VALUE *lookup_name(TOKEN * x, FRAME * frame){
    while(frame != NULL){
        BINDING *bindings = frame->bindings;
        while(bindings != NULL){
            if(bindings->name == x){
                return bindings->value;
            }
            bindings = bindings->next;
        }
        frame = frame->next;
    }
    return NULL;
}

VALUE *lookup_name_curr_frame(TOKEN * x, FRAME * frame){
    while(frame != NULL){
        BINDING *bindings = frame->bindings;
        while(bindings != NULL){
            if(bindings->name == x){
                return bindings->value;
            }
            bindings = bindings->next;
        }
        return NULL;
    }
}

VALUE *assign_to_name(TOKEN * x, FRAME * frame, VALUE * val){
    while(frame != NULL){
        BINDING *bindings = frame->bindings;
        while(bindings != NULL){
            if(bindings->name == x){
                bindings->value = val;
                return val;
            }
            bindings = bindings->next;
        }
        frame = frame->next;
    }
    printf("fatal: unbound variable!\n");exit(1);
}

VALUE *declare_name(TOKEN * x, FRAME * frame){
    BINDING *bindings = frame->bindings;
    BINDING *new = malloc(sizeof(BINDING));
    if(new != NULL){
        new->name = x;
        new->value = make_value_int(0);
        new->next = bindings;
        frame->bindings=new;
        return new->value;
    }
    printf("fatal: binding creation failed!\n");
}
```

```
61 VALUE *declare_func(TOKEN * x, VALUE* val, FRAME * frame){
62     BINDING *bindings = frame->bindings;
63     BINDING *new = malloc(sizeof(BINDING));
64     if(new != NULL){
65         new->name = x;
66         new->value = val;
67         new->next = bindings;
68         frame->bindings=new;
69         return new->value;
70     }
71     printf("fatal: binding creation failed!\n");
72 }
73
74
```