C Pointers & Structures

CS 350: Computer Organization & Assembler Language Programming

Lab 4, due Thu Sep 29

A. Why?

- Pointers let us share large memory objects without copying them.
- Structures give us a way to define data values that contain named components.

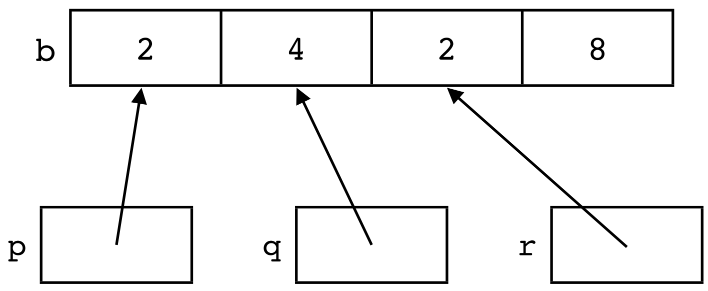
B. Outcomes

After this lab, you should be able to:

- Take a C expression or assignment that uses arrays and pointers and determine its value or action given a state of memory.
- Write simple C routines that take/modify structure arguments using pointers to the structure values.

C. Written Problems [50 points total]

1. [6 pts] Write some C declarations and code to establish the memory diagram below. (There are multiple right answers.) `p`, `q`, and `r` should be pointers to integers.



2. [14 = 7 * 2 pts] Using the memory diagram for Problem 1, answer the following question for each of the expressions below: Does it cause a compile-time warning or error (and if so, which one), or does it cause a runtime error (and if so, which one), or does it evaluate to true or false? [Hint: Write up your answer to Problem 1 as a

program; then try adding these expressions and compiling them.]

a. `p < q < r`

b. `p != r && *p == *r`

c. `q-b == &b[3] - &p[1]`

d. `p[1] == r[-1]`

e. `&r[-2] == &b[0]`

f. `q-p+q-p == q+q-p-p`

3. [15 = 6+9 pts] Consider the C declarations and code below.

   a. Draw a memory diagram that shows the state at position 1.

   b. Draw a memory diagram that shows the state of memory at position 2.

```
int b[4] = {12, 13, 14, 15};
int u = 20, v = 30, *x = &u, *y, *z;
y = &u;
z = &b[2];
// <----- Position 1
++ *x;  // (i.e., *x = *x + 1)
y = &v;
--z;
z[1] = 20;
// <----- Position 2
```

4. [15 pts] Consider the C declarations and code below; draw a memory diagram that shows the state at position 1. It's a bit tricky, but try to label all the different parts of the struct array fields (`x[0].a`, `x[0].b[0]`, … `x[0].str`, `x[1].a`, …).

```
struct struct_a {
    int a;
    int b[3];
    char *str;
};
typedef struct struct_a Struct_s;

Struct_s x[2], *p = NULL;
char *s1 = "hello";
int i;
for (i = 0; i < 3; i++) {
    x[0].b[i] = i*i;
    x[1].b[i] = -i;
}
p = &x[0];
p -> a = 12;
p -> str = s1;
p++;
p -> a = -23;
p -> str = s1;
int *q = &x[1].a;
// position 1;
```