C Pointers and Arrays

CS 350: Computer Organization & Assembly Language Programming

Due Fri Apr 18

A. Why?

- Pointers are an efficient way to share large memory objects without copying them.
- In C, pointers are used to simulate call-by-reference, and array references can be written as pointer dereferencing operations (and vice versa).

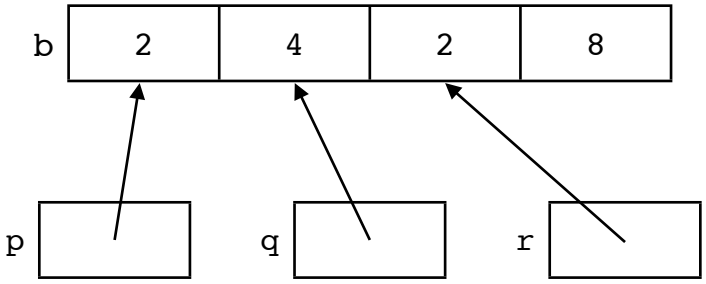
B. Outcomes

After this lecture lab, you should

- Take a C expression or assignment that uses arrays and pointers and determine its value or action given a state of memory.
- Translate between C code that uses array references and pointer references.

C. Written Problems [50 points total]

1. [12 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. [16 = 8 * 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?

   (a) `p == b`
   (b) `q == b+1`
   (c) `q == (&b)+1`
   (d) `*q == *(r–1)`
   (e) `p[1] == r[–1]`
   (f) `r-p == 2`
   (g) `p != r && *p == *r`
   (h) `q-b == &b[3] - &p[1]`

3. [10 = 2 * 5 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. [12 pts] The program in `Lab10_array_ptrs.c` declares an array `b` and a pointer `p` and manipulates them using array and pointer notation. Rewrite the code so that it swaps array notation for pointer notation and vice versa. Clean up the notation so that you use *thing* instead of `*&`*thing*.

