Lab 7: Structures, Pointers, and I/O

CS 350: Computer Org. & Asm. Lang. Pgm'g

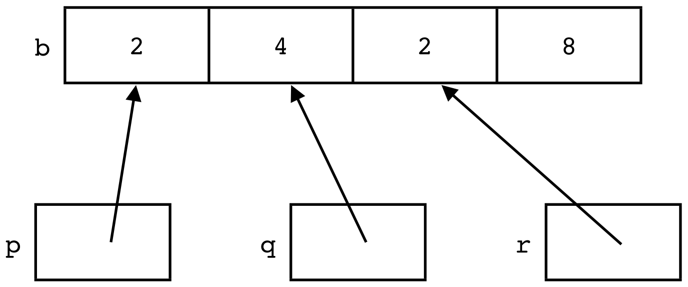
Due: Sat Apr 13, 11:59 pm

Problems [50 = 35 + 15 points]

This lab includes written problems and a programming problem. To submit copy your written and programming problem solutions to a folder, include your name in the folder, zip it, and submit the zipped file.

Written Problems [35 points]

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 following C declarations and code.

```
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
```

   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.

Programming Problem [15 points]

Your job is to write a simple C program to practice file I/O, arrays, and structures. Your program should

1. Print out your name and the fact that this is for Lab 7.
2. Open a file called Lab7.txt. (If the open fails, generate an error message and quit.)
3. Read a series of input; each input should consist of three numbers: an integer, a floating-point number, and an integer in hex.
4. Read the three numbers into elements of an array of structures.[1]

```
struct Lab7_data {
    int x;
    float y;
    int z;
};
struct Lab7_data data[100];
```

   If `i` is the index into the array, then read into `data[i].x`, `data[i].y`, and `data[i].z`.
5. Go through the read-in part of the array in reverse order. Print the three numbers of the structure for each element of the array.

Opening and Reading a File

- Here's the basic technique for opening a file for reading. The `exit(EXIT_FAILURE)` stops the program with an error. The test `!datafile` returns true when the `datafile` pointer is null, which happens when the file opening fails.

```
#include <stdio.h>
#include <stdlib.h>       // For error exit()

int main(void) {
    char *datafile_name = "Lab7.txt";
```

[1] We can live dangerously and just assume we don't run out of array elements. Or you can keep track of the number of elements read in and stop when the array is full.

```
FILE *datafile = fopen(datafile_name, "r");
if (!datafile) {
    printf("; open failed; program ends\n");
    exit(EXIT_FAILURE);
}
```

- To read from the data file, use the `fscanf` input routine; it's just like `scanf` but takes the `FILE` pointer as an additional argument, before the format part of the call. For example, to read an integer into variable `x`, you use

  ```
  fscanf(datafile,"%d", &x);
  ```

  (Your `fscanf` will be more complex, since you're reading in three pieces of data, into the fields of a structure that's an array element.)
- Close the file once you're done reading into it, using `fclose`(`datafile`).