

Remarks: Keep the answers compact, yet precise and to-the-point. Long-winded answers that do not address the key points are of limited value. Binary answers that give little indication of understanding are no good either. Time is not meant to be plentiful. Make sure not to get bogged down on a single problem.

PROBLEM 1 (48 pts)

(a) Consider the code snippet

```
int x, *y, *z; z = &x; *z = 3; *y = 5; x = 7;
```

The code will not cause compilation error but trigger a run-time error. Specify which statement contains a run-time bug, what the run-time error is called, and why it is generated.

(b) Consider the code snippet

```
int x, y, *z; x = 1; printf("%d\n", x); y = 2; printf("%d", y); *z = 3; printf("%p", z);
```

What is likely to be output (and visible) on the display? Explain your reasoning.

(c) Given a 1-D array, `char s[10]`, why is `*s` the same as `s[0]`? Base your explanation on the discussion in class regarding memory layout of 1-D arrays where we ignored an optimization performed by gcc. Why is `*s+2` not the same as `s[2]`?

PROBLEM 2 (34 pts)

(a) Consider function `main()`

```
int main() { char v[10]; scanf("%s", v); printf("%s", v); }
```

which reads a string into char array `v` which is then output to stdout. Explain why this code is flawed in the sense of having a nascent bug and what may happen at run-time for some input. Is `printf()` likely to succeed by printing `v` to stdout? What is a simple fix to the above problem that continues to use `scanf()` to read a string into `v` from stdin?

(b) The library function, `int fgetc(FILE *fp)`, reads a character (i.e., byte) from a file pointed to by file pointer `fp`. Why is the return value of `fgetc()` of type `int` and not `char`? Write a snippet of code that reads the content of a file pointed to by `fp` (i.e., file has been successfully opened) and determines if it's an ASCII text file. Do not use bit processing techniques in your solution. What is the logic behind your code?

PROBLEM 3 (18 pts)

A variation of Problem 2(b), use bit processing techniques (i.e., shifting bits, AND'ing with a mask) discussed in class to determine if a file pointed to by `fp` is not an ASCII text file. As before, assume that a file has been opened successfully and pointed to by `fp`, and the content of the file is read byte-by-byte using `fgetc()`.

BONUS PROBLEM (10 pts)

Suppose function

```
int abc(void) {static int r = 10; r++; return r++; }
```

is called twice by `main()`. What values are returned by `abc()`, and why?