CS 240 (Park) **Final** **Aug. 3 (Wed.), 2022**

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 (45 pts)

(a) Which statements in the code

> typedef struct friend { char *nickname; unsigned int year; } friend_t; int main() { friend_t *amigo; amigo–>year = 2017; strcpy(amigo–>nickname, ”fish”); }

are problematic, likely to trigger segmentation fault? Augment the code by adding calls to malloc() so that the bugs are fixed.

(b) Explain the difference between fun1 and fun2 which are declared as char *fun1(char *) and char (*fun2)(char *), respectively. Code a function fun3 that takes a string as argument and returns the last character of the string. You may assume that the string is of length at least 1 (not counting EOS).

(c) Suppose a user enters the command, % /bin/cp file1 file2, using a shell to copy the content of file1 to file2 on one of our lab machines. From the viewpoint of the shell, from where does it read its input ”/bin/cp file1 file2”? From the viewpoint of the app /bin/cp which is coded in C, how does it access its input which specify the names of two files whose content is to be copied? Before calling execv() what must the shell do to prepare the arguments of execv() so that /bin/cp has access to the two file names?

PROBLEM 2 (30 pts)

(a) Code a function, unsigned int countdbl(long), that takes a number of type long as input, counts the number of 0s in the bit representation of the input, and returns 0 if the count is an even number, 1 if odd. Use bit processing techniques to solve the problem.

(b) gcc on our lab machine, by default, will insert code to detect stack smashing at run-time. What does gcc's code try to prevent from happening? In the case of reading input from stdin (or file), what is a common scenario and programming mistake that can lead to stack smashing? Provide an example using scanf() (or fscanf()). What is sound programming practice that prevents stack smashing?

PROBLEM 3 (25 pts)

Code a function that takes variable number of arguments, double multnums(char *, ...), multiplies them and returns the result as a value of type double. The fixed argument is a string that specifies how many arguments follow and their type (integer 'd' or float 'f'). For example, in the call multnums(”dffd”, 3, 88.2, -100.5, 44), the format string ”dffd” specifies that four arguments follow where the first character 'd' means the first argument in the variable argument list is of type integer, the second and third 'f' of type float, and the fourth 'd' of type integer. Forgo checking for errors and ignore header files. What would happen in your code if multnums is called as multnums(”dffd”, 3, 88.2, -100.5, 44, -92, 65)? What about multnums(”dffd”, 3, 88.2, -100.5)? Explain your reasoning.

BONUS PROBLEM (10 pts)

Suppose an ASCII file contains lines where each line is a sequence of characters ending with '\n' but for the last line which ends because the end of file is reached. The goal of main() is to read and store the lines of the ASCII into a variable, char **x, where malloc() is used to allocate just enough memory to store the content of the file. Using only basic file I/O operations discussed in class, describe in words how your code would work to accomplish this task. Be detailed in how the arguments of malloc() are determined to store the file content in x.