1. Yes, there is a memory leak in this program. Execl will load the kernel’s address space with a new program – in this case, bash. Because of this, all code below execl() will not get executed. For this reason, the memory allocated by *cmd* will never get deleted, and there will be a memory leak. For this reason, it is bad practice to dynamically allocate a character array in in a child process, when planning to pass that character array on to another process.

To decide whether or not this leak is dangerous really depends on the context of the rest of the code. If this process gets called a single time, this one memory leak will not be too dangerous. If, however, this is a process that will be executed many times, allocating 128 bytes repeatedly without deallocating it will run up the system’s memory.

1. Yes, they both produce the same output. The same processes get called at execution.
2. While performing the “exec” command, the first and second arguments are often the same, but not always. The first argument is the binary to load for execution. The second is the name of the process. In the case of the lab, these were the same; however, it was not entirely necessary. Take, for instance:

#define BASH\_EXEC “/bin/bash”

.

.

.

If (execv(BASH\_EXEC, “BASH\_EXEC”, “-c”, cmdbuff, (char\*) 0) < 0) {…}

“BASH\_EXEC” encompasses the entire path to the process being executed; however, this is redundant and could have been replaced with simply:

“bash”

1. If you attempted to read from an empty pipe, your process would do one of two things. If the “write” end of the pipe was still open, the “read” end would wait for some input. If the “write” end is not open, the program would throw an error.
2. There is a limit to how much data can be buffered in the pipe by the producer without affecting the producer’s operations, and it is system-dependent. Linux, for instance, has buffers of sizes 4k to 64K depending on the version. If the pipe buffer is full, however, the data is not simply lost; rather, the consumer will block until space is available.