Consider the three command lines:

% gcc my\_prog.c

% ./a.out < in1 > out1 & ./a.out < in2 > out2

% ./a.out < in3 | ./a.out > out3

1. In response to the second command line, are there one or two programs running? Are there one or two processes running?

One program = my\_prog.c  
 Two processes = each instance of one program

2. In response to the third command line, are there one or two programs running? Are there one or two processes running?

One program, two processes

3. What is the difference between the second and third command lines?

2: separate input and output for processes

3: piping output of first process as input of second process

Kernel mode / User mode. Circle **one or both** of K and U, as applies.

4. K / U In this mode add instructions can be executed.

5. K / U In this mode only a subset of the instructions can be executed.

6. K / U In this mode only a subset of the physical memory addresses can be accessed.

7. Consider the changes in the memory image of a process during a buffer overflow attack, as illustrated below. Identify at least one way in which the attack can be prevented. (diagram source: stackexchange)

Buffer overflow example in stages.
Stage a: stack layout before function call.
Stage b: stack layout after function call with return address to main program. There is a small buffer allocated as a local variable for the function.
Stage c: stack layout after buffer overflow, where the return address has been replaced with a pointer to code that has been read into the buffer.

execution starts in main() main() calls A() buffer overflow after a call to strcpy()

returns to A(); the return address to

main() has been overwritten

Can be prevented by marking stack as non executable