1.
    a. The issue with the previous code is that for memset to work it needs to be provided a pointer, to do this we use the ampersand. The errno is also unnecessary as memset returns a void.
    b. This code is also not correct as the AI forgot to subtract one from the length when checking for and removing the '\n' character. It also does not correctly remove '\n', to do that I used ' \t'.
    c. This code does not print to stdout, perror prints to stderr. This can be fixed by using fprintf while using errno.
    d. The function returns the tokens correctly but if the string exceeds the global limit the function will fail.
    e. The first example used is between the printf, strcpy, and atoi functions and the write system call. Here it explains that the printf function utilizes the write system call to access the kernel to output a string while strcpy and atoi do not use any system calls to complete their jobs. The other example is between sbrk(2) and malloc(3). With this example it is explained that sbrk(2) is not a general purpose memory manager, it only increases or decreases the address space of the process by a specified number of bytes. Malloc functions tell a process how to manage the memory allocation in its given memory space.
2.
    a. The code provided almost works but has two mistakes. The first is that it got 'w' and 's' backwards with 'w' causing the program to sleep when that should be caused by 's'. The other problem is that it provides the wrong value to waitpid(). Status should be &Status to provide a pointer for the function instead of the value itself.
    b. WIFEXITED is a macro not a function call, it is part of the library "#include <sys/wait.h>" and uses #define.

Experiment 1: a1p2.w

	PID	PPID	State	CMD
parent	3002244	2971194	S	./a1p2 w
child	3002245	3002244	S	/bin/sh ./myclock out1
grandchild	3002458	3002245	S	sleep 2

Experient 2: a1p2 s

	PID	PPID	State	CMD
parent	3004280	1	S	/bin/sh ./myclock out1
child	3004441	30047280	S	sleep 2