1. Answer: string of 65 ‘a’s,
   Looking at the source code, I see that it creates a variable, “modified”, and then “buffer”, which is 64 bytes. To obtain the correct output, I just have to find a way to change “modified”. Since my input will go into “buffer”, I know that I will be able to overflow it since it uses the function “gets()”, which doesn't use bounds checking. This means that if I insert a value larger than 64 bytes, I will be overwriting “modified”. I tested this by inputting a string of 65 of the character ‘a’., and the program output the message: “you have changed the ‘modified variable’”.

2. Answer: (‘a’*64)+’dcba’
   This code is similar to the previous one except for the fact that rather than just changing the value ‘0x61626364’. Translated from hex, this value is equal to the string ‘abcd’/ This means that when I overflow my buffer, I just have to insert that value at the end, but in reverse, since it will be read in little endian. Taking all of this into account I insert the argument as a python script(python -c “print ‘a’*64+’dcba’”). This sets the second argument to 64 ‘a’s followed by ‘dcba’. When I enter this argument, I receive the output: “you have correctly got the variable to the right value”

3. Answer: export GREENIE=$(python -c “print ‘a’*64+’\x0a\x0d\x0a\x0d’”)
   This code is similar to the previous two except for the fact that the value is obtained from an environment variable “GREENIE” rather than simply input by the user. The buffer will be set to whatever the “GREENIE” variable is set as, so we first want to start with ‘a’*64 to fill in the buffer as usual. Next we see that “modified needs to be set to “0x0d0a0d0a”. This isn't as simple as before because these values are equivalent to a carriage return and newline, which we can't just input like we could with the character strings. To account for this, we can set the variable as a python string, starting with ‘a’*64 of course. Then, since python can output hex values if the prefix ‘\x’ is used, we will add the reverse of the required value, because of little endian, which will be ‘\x0a\x0d\x0a\x0d’. Once “GREENIE” is set, simply run ./stack2. This result in the output: “you have correctly modified the variable”.