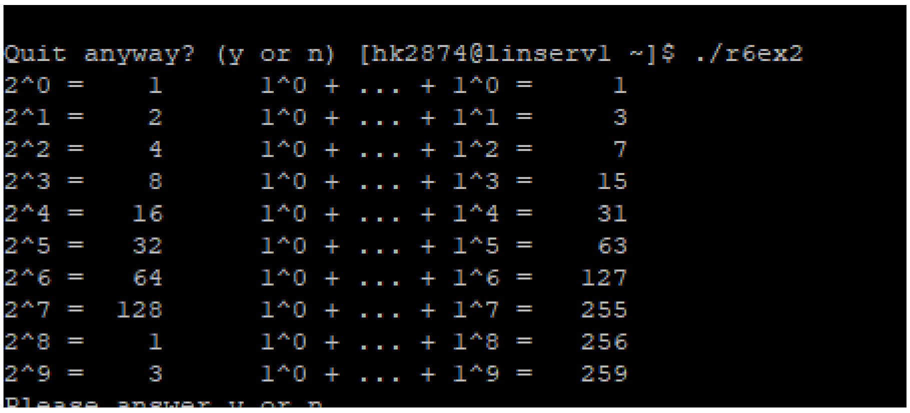
My output of the file did not match the readme file.
It looked like this and I answered questions based on my output.



```
Quit anyway? (y or n) [hk2874@linserv1 ~]$ ./r6ex2
2^0 =    1      1^0 + ... + 1^0 =     1
2^1 =    2      1^0 + ... + 1^1 =     3
2^2 =    4      1^0 + ... + 1^2 =     7
2^3 =    8      1^0 + ... + 1^3 =    15
2^4 =   16      1^0 + ... + 1^4 =    31
2^5 =   32      1^0 + ... + 1^5 =    63
2^6 =   64      1^0 + ... + 1^6 =   127
2^7 =  128      1^0 + ... + 1^7 =   255
2^8 =    1      1^0 + ... + 1^8 =   256
2^9 =    3      1^0 + ... + 1^9 =   259
Please answer y or n
```

1. What are the memory addresses at which both arrays are stored? Specify the instruction that you used to figure that out.

   Hint: The address you are after is the address of the zero'th index in the array (of course, there are many ways of printing that value).

`(gdb) print &vals[0]` which gave 0x602010
`(gdb) print &partial_sums[0]` which gave 0x602030

2. Once the loop on lines 18-22 completes, what are the values saved in the `vals` array? Specify the `gdb` instructions that you used. List all the values in the array.

   Hint: you can set a breakpoint on line 23 and just let the program continue untill it hist that breakpoint - this way you do not have to manually step through every iteration of the loop.

`(gdb) break 23`
`(gdb) run`
`(gdb) print *vals@10` which printed out
`{1, 2, 4, 8, 16, 32, 64, 128, 256, 512}`

3. Once the loop on lines 26-30 completes, what are the values saved in the `vals` array? what are the values saved in the `partial_sums` array? Specify the `gdb` instructions that you used. List the values in both arrays.

Are the numbers stored in the two arrays what they supposed to be?

`(gdb) break 32`
`(gdb) continue`
`(gdb) print *vals@10` which printed out
`{0, 0, 0, 0, 0, 0, 0, 0, 0, 0}`

Yes the numbers stored in the two arrays are what they are supposed to be.

4. Go back to your answer to question 1. What is the difference between the two memory addresses (in decimal)?   Does this make sense?
6299696- 6299664 = 32

No. Since one integer has 4 bytes and there are 10 elements in each array, the difference between the two memory adresses should be at lest 40.

5. Run the program to line 45 and examine the values of the `partial_sums` array? Are those the values that are printed when the program is actually executed?
Show the values actually stored in the array and the values that are printed when the program is executed.

On line 45, the values actually stored in the array are
{1, 3, 7 , 15, 31, 63, 127, 255, 256, 259}

However, it is not the values that are printed when the program is actually executed.

6.
In the second breakpoint(at line 34), when I executed
x/10dw vals
x/10dw partial_sums

there is an overlap in memory that those two arrays occupy at 0x602030. So the last 8 bytes of the vals array overlapped with first 8 bytes of the parital sums array.

After each line, when display the content of `partial_sums` array after line 45, the content of the array did not change.

Actually, regarding the output that I had, the content that got changed was at the loop that initialized the array partial_sums to zeros, because there was a overlap in adresses which deleted the initial correct last values in the array val.