**binary1:**

1. How big is the .text section (1 pts)?
2. What is the starting address of main (1 pts)?
3. How many, if any, custom functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
4. How many, if any, system functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
5. For each custom function in the binary, provide a minimum of 2-3 sentence describing what the function does. Your description should include any functions called, variables created or modified, and the purpose of the function. Include main in your answer. (3 pts)
6. Provide a minimum of 2-3 sentences describing what the entire program is doing. Include why the functions you’ve listed in questions 3&4 are necessary to the program’s function. (3 pts)

s

**1.** .text size: 0x1d2

**2.** main() start address: 0x5756513d

**3.** no custom functions exist in this binary (other than main)

**4.** One function call to puts(), called at main+25 (0x5756514e).

**5.** main(): this is a really simple program that just calls puts, passing in only a null terminated string (in this case “hello world”), and then returning.

**6.** the program is a hello world program. It just prints “hello world” to the screen and then returns 0

**binary2:**

1. How big is the .text section (1 pts)?
2. What is the starting address of main (1 pts)?
3. How many, if any, custom functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
4. How many, if any, system functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
5. For each custom function in the binary, provide a minimum of 2-3 sentence describing what the function does. Your description should include any functions called, variables created or modified, and the purpose of the function. Include main in your answer. (3 pts)
6. Provide a minimum of 2-3 sentences describing what the entire program is doing. Include why the functions you’ve listed in questions 3&4 are necessary to the program’s function. (3 pts)

**1.** .text size: 0x252

**2.** main() start address: 0x58565194

**3.** one call to populateArray() at main+30 (0x585651a4)

**4.** one system function, with two calls to puts(), first called at main+55 (0x585651bd)

**5.** populateArray(): this function accepts two arguments, and creates two local variables. I did not realize what the first argument was used for until working on binary3, but it appears that the first argument is a pointer to the address space being written to The second argument determines the number of iterations that the loop inside the function will run. The first iteration overwrites the contents of virtual memory address 0x0 (1byte not 1word) with a locally defined variable. Each subsequent iteration of the loop overwrites the next byte of memory, 0x2,0x3, and so on until the end of the loop (as defined by the second argument). As far as I can tell, the stuff being written to memory is gibberish. *True to the name though, even if the content is gibberish, it is populating an array of 1byte elements with data.* This function always returns 0.

main() initializes a local pointer variable to zero, passes the values 0x14 and 0xffffcff8 to apopulateArray() and then compares the return value of the function to 0. The flags triggered by that comparison are checked against an if/else jump structure, wherein a return of 0x0 prints “failure” using puts(), and a return of nonzero results in printing “success” using puts instead. After the print call, main returns.

**6.** This program overwrites a section of memory with seemingly meaningless values. Maybe they are opcodes, but they seem like a bunch of escaped characters.

**binary3:**

1. How big is the .text section (1 pts)?
2. What is the starting address of main (1 pts)?
3. How many, if any, custom functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
4. How many, if any, system functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
5. For each custom function in the binary, provide a minimum of 2-3 sentence describing what the function does. Your description should include any functions called, variables created or modified, and the purpose of the function. Include main in your answer. (3 pts)
6. Provide a minimum of 2-3 sentences describing what the entire program is doing. Include why the functions you’ve listed in questions 3&4 are necessary to the program’s function. (3 pts)

**1.** .text size: 0x292

**2.** main() start address: 0x30101b3

**3.** 2 custom functions not counting main  
populateArray() is called @ 0x595651df

getRandomLetter() is called @ 0x595651f9

**4.** four system functions  
\_\_vdso\_time() is called @ 0x595651c2

srandom() is called @ 0x595651ce

rand() is called @ 0x5956518c

puts is called @ 0x5956520e

**5.** populateArray(): - discussed in bunary2, I have confirmed it this function is identical to the one called in binary2, the iteration count is identical as well, the only thing that has changed is the value of the strange first argument, which points to the head of the string/array/memory space to be written.

getRandomLeter(): takes a char \*string as the first argument, and the length of the array/string as the second argument. Right off the bat, it calls rand(), and runs a modulo operation against the second argument to generate an index of for the char\*string. The function returns a pointer to the randomly selected index in the eax register.

main(): calls the functions responsible for opening the time library, seeding the random number generator with the time, and generating an array of garbage data. The function then checks a variable on the stack to ensure it is not null, if I is null it prints “FAILURE” to the screen. If it is not null, then selecting a single element of that random array, and doesn’t appear to do anything with it. The function then returns

**6.** aside from re-iterating what main is doing, I’m a little lost on how to answer this. The program is generating an array, and then selecting an element from it at random. The rand function is necessary to generate the randomness, however it becomes predictable if you fail to provide a unique seed each run. To that end, the srand and vdso\_time functions are necessary. The two custom functions facilitate the creation of the array (which may be superfluous since you could have a prepared array… the array itself is not seeded with random values) and the selection of the ‘random element.’

**binary4:**

1. How big is the .text section (1 pts)?
2. What is the starting address of main (1 pts)?
3. How many, if any, custom functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
4. How many, if any, system functions exist in the binary? List their names and the address the function is first called at, omitting duplicates (do not include main.) (1 pts)
5. For each custom function in the binary, provide a minimum of 2-3 sentence describing what the function does. Your description should include any functions called, variables created or modified, and the purpose of the function. Include main in your answer. (3 pts)
6. Provide a minimum of 2-3 sentences describing what the entire program is doing. Include why the functions you’ve listed in questions 3&4 are necessary to the program’s function. (3 pts)

**1**. .text size: 0x482

**2**. main() start address: at 0x4010397

**3**.custom functions: 2

populateArray() @ 0x5a5653c8

printArray() @ 0x5a5653dd

sortArray() @ 0x5a5653ef

**4.**System functions: 4

\_\_vdso\_time() @ 0x5a56539f

srandom() @ 0x5a5653ab

printf() @ 0x5a565365

putchar() @ 0x5a56537e

**5.** I’m calling it quits here so I can still submit this on time. Didn’t leave myself enough time to get everything done. Lesson learned. I could probably throw a guess together about what is going on in the program just based on the function names, but I have not honestly read into it beyond that, so I might not have everything in order… like clearly the sortArray function sorts the array it is passed… The only thing I know about it though is that it doesn’t call any other functions from within, I haven’t read it and I don’t have the time…. Darn.

**6.**