Question 1)
Sql injection.
Expecting something like “if ‘input’ == user.password ”
So in the password box, enter

With the line:
‘ OR 1==1; --
This line closes the input, with the ‘, then OR 1==1 makes the statement always true, and finishes with a line comment, to avoid errors.

Part 2)
Description:
Using the random number generator and input with the seed, the idea is that you run the random memory commands until there’s an error, which should be expected eventually.

Alternatively, you input a random string of text, which is converted to a number and this is used as input seed for the random number generator.

Bug1:
Using my random number generator on the original code, with the input of 0 gave me pointers to my first bug in sp_realloc.

The line nested between the 2 printf’s in sp_realloc() was causing errors. Unfortunately, I was not able to understand what os does, and hence how to fix the bug.

```
if (chunk != NULL) {
    uintptr_t id = ((uintptr_t) chunk - ARENA_BASE) / SUBARENA_SIZE;
    if (id < NUMCHUNKSHIFT)
      os = (1ULL << (id + MINCHUNKSHIFT)) & (!((size & 0xff)^0x37)-1);
    else
    {
      if (((uintptr_t)chunk & PAGEMASK) == 0ULL)
      {
        printf ("{1}\n");
        os = (*(uint64_t *)((uintptr_t)chunk - PAGESIZE)) << PAGESHIFT; /*
this is causing bugs */
        printf ("{2}\n");
      }
      else
      {
        abort();
      }
    }
  }
```

As I was unable to find the solution to the bug, I commented this line out. As a result, free would no longer work. I'm assuming this was partially due to this line:

```
if (size <= os)
    return chunk;
```

Meaning that data that was supposed to be dealt differently (e.g it was meant to be freed and would be freed with normal implementation, but now isn't).

Bug 2:
Using the input 13 in my random number driver, gave me my next bug.

```
void *sp_calloc(size_t nmemb, size_t size) {
  void *rv = sp_malloc(nmemb * size);
  printf ("{11}\n");
  bzero(rv, nmemb * size);
  printf ("{12}\n");
  return rv;
}
```

When calloc is called, something fails inside calloc. It is related to bzero() function apparently, but it also might be related to the read/ write privileges of the data, as calloc is the only function that actually writes to memory in its function call and my driver function doesn't fuzz properly with random inputs.