- **Create a struct:**
  - That has a string to store the result of the reading. Called ***str***.
  - An ***int*** to store the File Descriptor. Called ***fd***

- **Create the get_next_line function:**

  VARIABLES
  - Create a static of the above struct, called ***retstore*** (for now)
  - A buffer of [BUFF_SIZE + 1], called ***buff*** (for now)
  - As well as an ***int*** to store the number of bytes read, called ***ret*** (for now).
  - A string to act as a temp holder of some result, called ***temp***.

  CODE
  - First check if ***fd*** is valid and also if the variable ***line*** is not NULL
  - Secondly check if my static ***struct*** is NULL. If NULL allocate memory (using ***malloc***) for it and assign the necessary information to the relevant places like the File Descriptor. Perhaps by even creating a function for this, I will call it ***ft_makestruct*** (for now)
  - I then execute the read function as follows:

    ret = read(fd, buff, BUFF_SIZE)

    nb.: executed in a while loop to compensate for situations where a BUFF_SIZE is less than the required length.
  - Inside the while loop:

    buff[ret] = '\0';

    retstore->str = (a function to join the contents of retstore's str with whatever is inside buff)
  - Outside the while loop:

    I check if my ***ret*** is equals to -1, if it is I return -1 to signify an error during the read process.

- **Then using my static variable *retstore* as well as the parameter line for the next step:**
  - I run the ***ft_strchr*** function on ***(retstore->str, '\n')*** to find the pointer to that position, and store it into temp.
  - I then check if my temp is equals to NULL, which will mostly happen if the character wasn't found.
  - If it's not equals to NULL:

    ****temp = '\0'*** *//making sure the string is completed*

    ****line = ft_strdup(retstore->str);*** *//copies the contents of **retsore's** string into line*

    retstore->str = ft_strdup(temp + 1); *//copies the contents of temp into **retstore's** string*

    return (1); *// to signify a successful read.*
  - In a situation where my retsore's string is empty, that being that ***ft_strlen(retstore->str)*** is less than or equals to ***0***:

    ****line = ft_strdup(retstore->str);***

    ****(retstore)->str = '\0';*** *//the character at said position becomes a null terminator*

    return (1); *//signifying a successful read*
  - Then return zero at the end which signifies nothing has been read.

nb.: might consider turning the whole of third bullet into a function to stick to the norm.