OS ASSIGNMENT-1 LOADER:

Group member names:

Parsh Jain (2023368), Aakash Chityala (2023005)

Explanation of our code:  
Our assembler code has been written in such a manner following all steps as given in the provided pdf file. We have done the following things:

* For the loader cleanup function, we cleared and freed the pointers for program and elf header along with any space allocated in memory. This extra allocated space was cleared using munmap() function.

Load and running of elf file:

* We first opened the file in executable format and then found its file size using lseek function.
* Then we allocated memory for copying binary data using read function where we stored the content of ELF header in the heap memory pointed by ehdr
* We did a similar task when storing content of the program header in a heap memory pointed by program header offset.
* Then we iterated in the program header table to find those segments which were of the type of PT\_LOAD. Upon finding such occurrences, we stored it in a virtual memory using mmap function.
* We also added another condition to check to see whether the obtained PT\_LOAD type segment also contained the e\_entry offset as mentioned in the instructions.
* Since it may be possible that e\_entry offset may be at the beginning of the segment, we created relative\_dist variable to ensure that our actual entry is placed at a similar point in the virtual memory as the one found relative to its position in the program header segment
* After storing the virtual address of \_start function in variable actual\_entry, we then typecasted it into function pointer and simply called it and print the returned value.
* Finally after running and loading our ELF file, we got our desired output and at the end performed a loader cleanup to free and clean the used space.
* We also implemented error handling in such a manner such as to see whether we were getting the desired return statement, and if not then the code would show an error. For example in case of lseek() and read() function the returned value had to be greater than 0 for it to have completed the assigned task, and in case of file opening it should have been greater than or equal to 0.

With Bonus:

* First of all we created a top-level Makefile which will call the inner directory Makefile in the test, loader and launcher.
* The next step was to provide valid commands in each of the inner directory Makefiles, which will automatically execute and create their respective object files.
* As Loader.c file has to be same as in the without-bonus part, so there were no changes made in it.
* In the launch.c, the first job it does is to validate the elf file, and if it is then to pass it to loader.c for execution.
* Moving the lib\_simpleloader.so and launch executable into the bin folder.
* Finally executing the make command in the top-level Makefile and running the ./launch ../test/fib command to get the output.

GitHub Repository Link:

<https://github.com/solardroids/OS-Assignment-1-Simple-Loader>

Contributions:

Aakash:- Brainstorming, Code writing and execution

Parsh:- Brainstorming and debugging