CS 2263 - FR01A

Lab 4

By Ngoc Phuong Anh Nguyen - 3712361

October 2021

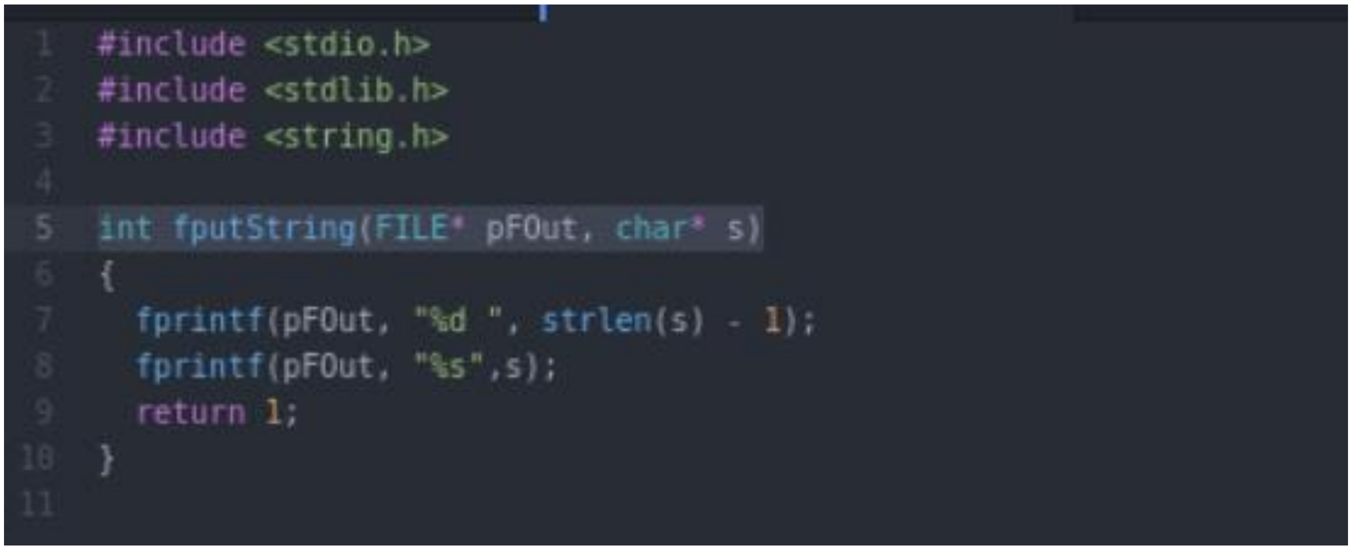
Exercise 1:

Question 1: Write the function to do this:

```
int fputString(FILE* pFOut, char* s);
```

that writes the string to the file, preceded by the length of the string as an integer, e.g.

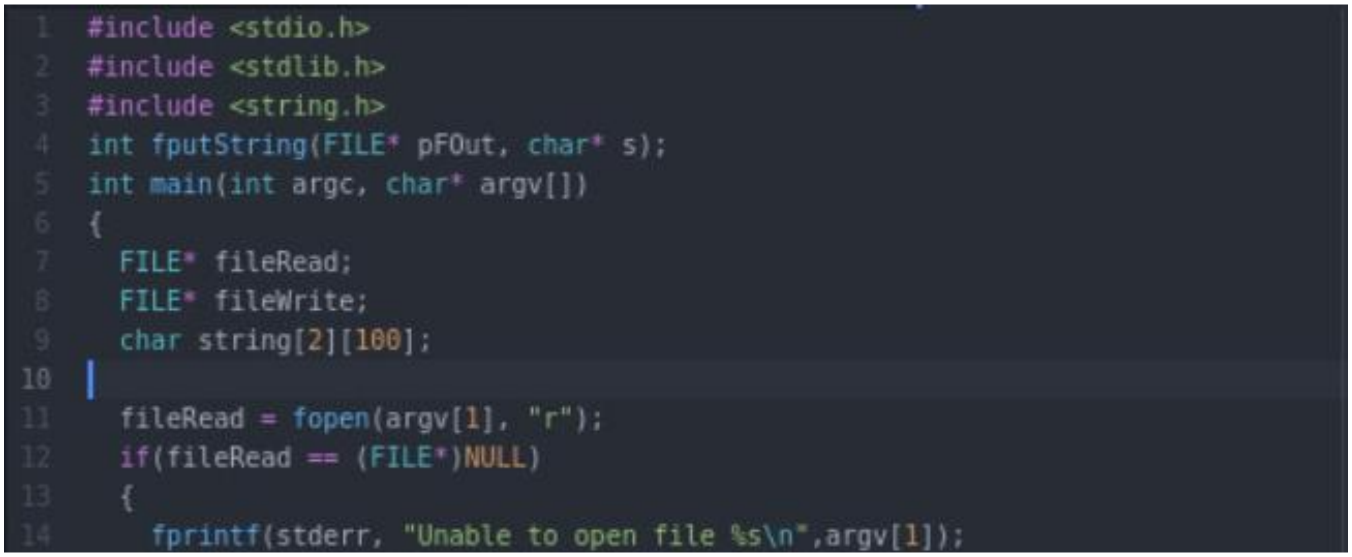
```
8 I love C!
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int fputString(FILE* pFOut, char* s)
{
  fprintf(pFOut, "%d ", strlen(s) - 1);
  fprintf(pFOut, "%s",s);
  return 1;
}

```

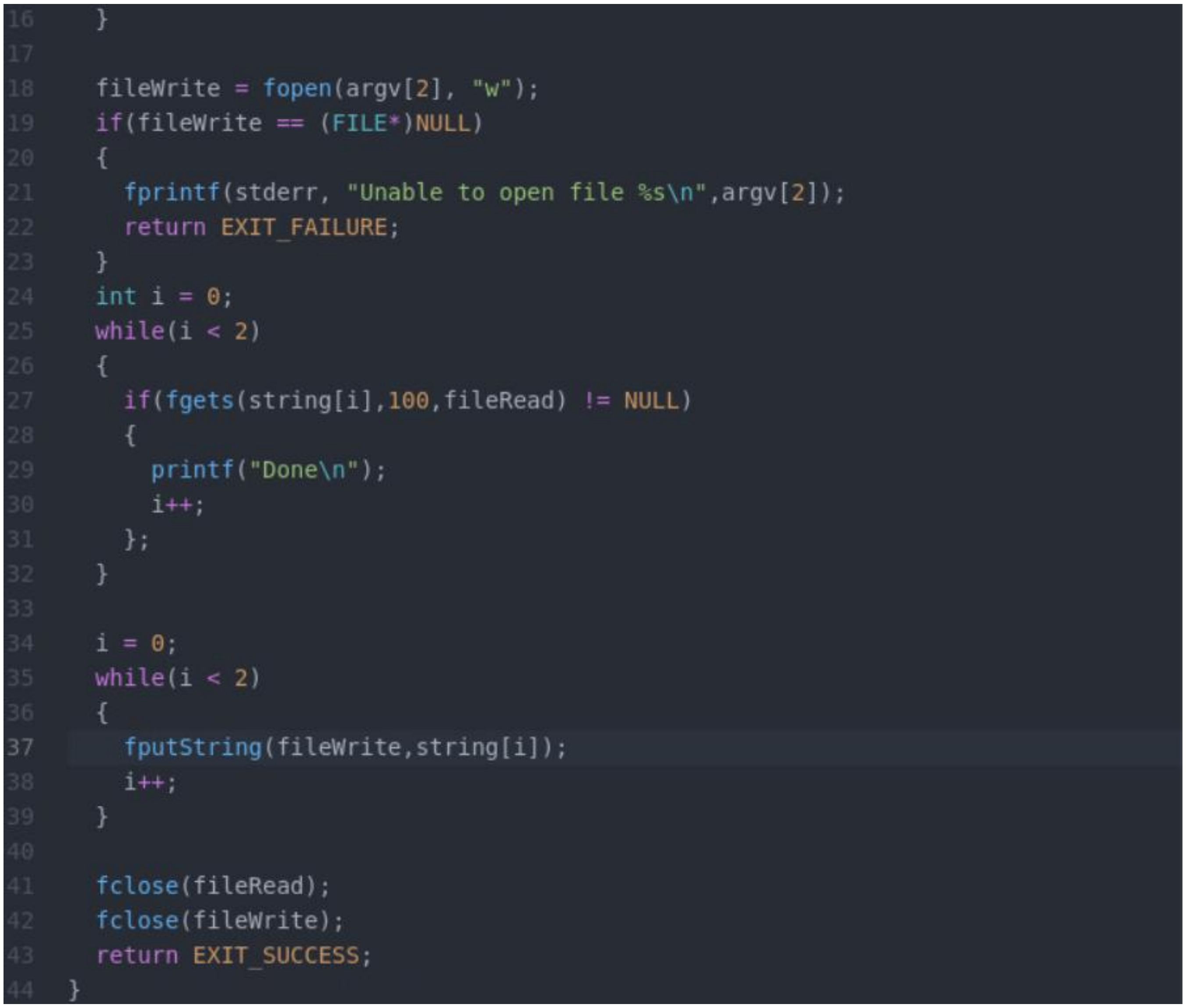
Figure 1: Source Code of fputString() function

Question 2: Now write a test driver program that reads two strings (assume less than 100 characters each) from standard in using fgets() and writes them to an open file (file name from the command line arguments) using your function. Of course, you should add this as a new target in your Makefile before you get started.



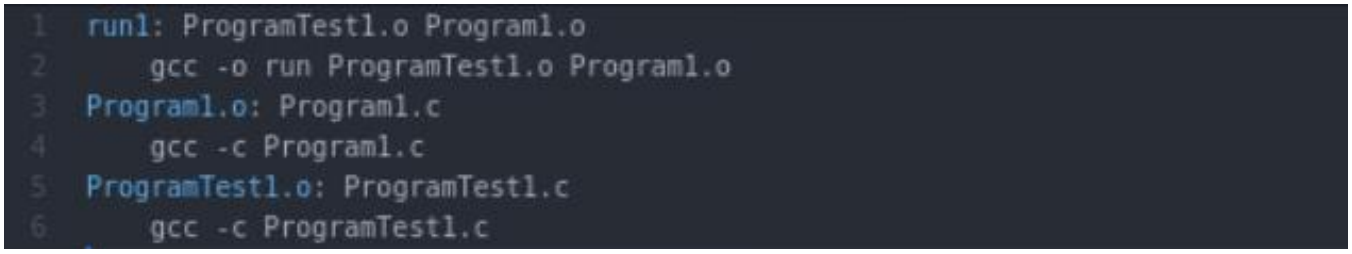
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int fputString(FILE* pFOut, char* s);
int main(int argc, char* argv[])
{
  FILE* fileRead;
  FILE* fileWrite;
  char string[2][100];

  fileRead = fopen(argv[1], "r");
  if(fileRead == (FILE*)NULL)
  {
    fprintf(stderr, "Unable to open file %s\n",argv[1]);
```

Figure 2: Source Code of Test Driver Program



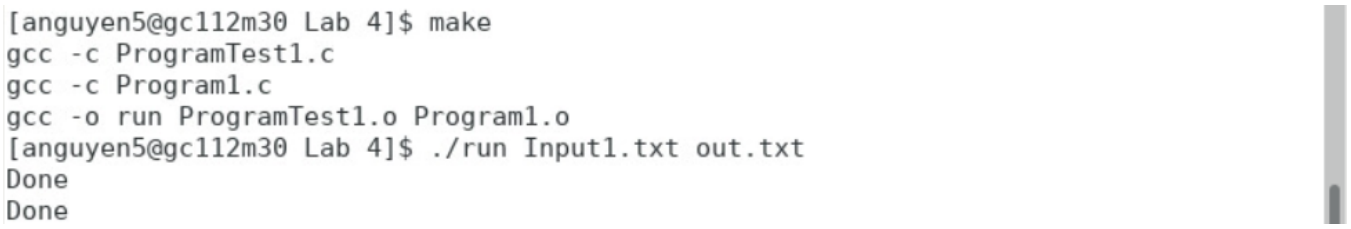
```c
  }

  fileWrite = fopen(argv[2], "w");
  if(fileWrite == (FILE*)NULL)
  {
    fprintf(stderr, "Unable to open file %s\n",argv[2]);
    return EXIT_FAILURE;
  }
  int i = 0;
  while(i < 2)
  {
    if(fgets(string[i],100,fileRead) != NULL)
    {
      printf("Done\n");
      i++;
    };
  }

  i = 0;
  while(i < 2)
  {
    fputString(fileWrite,string[i]);
    i++;
  }

  fclose(fileRead);
  fclose(fileWrite);
  return EXIT_SUCCESS;
}
```

Figure 3: Source Code of Test Driver Program



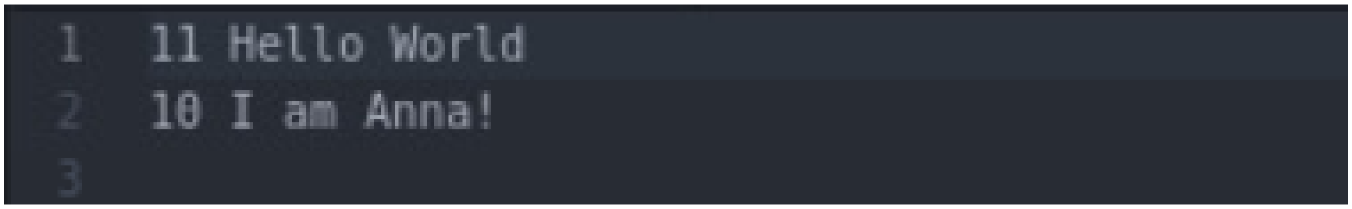
```makefile
run1: ProgramTest1.o Program1.o
	gcc -o run ProgramTest1.o Program1.o
Program1.o: Program1.c
	gcc -c Program1.c
ProgramTest1.o: ProgramTest1.c
	gcc -c ProgramTest1.c
```

Figure 4: makefile for Question 1



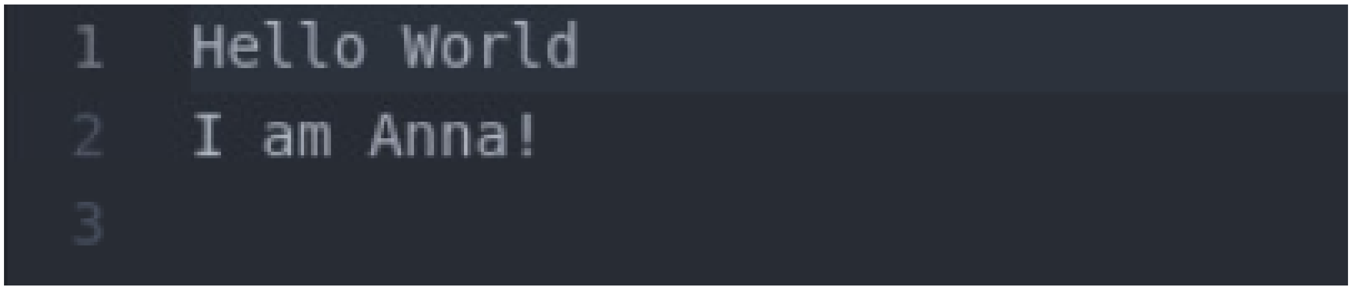
```
[anguyen5@gc112m30 Lab 4]$ make
gcc -c ProgramTest1.c
gcc -c Program1.c
gcc -o run ProgramTest1.o Program1.o
[anguyen5@gc112m30 Lab 4]$ ./run Input1.txt out.txt
Done
Done
```

Figure 5: Output using makefile



```
11 Hello World
10 I am Anna!
```

Figure 6: Output in the out.txt file



```
Hello World
I am Anna!
```

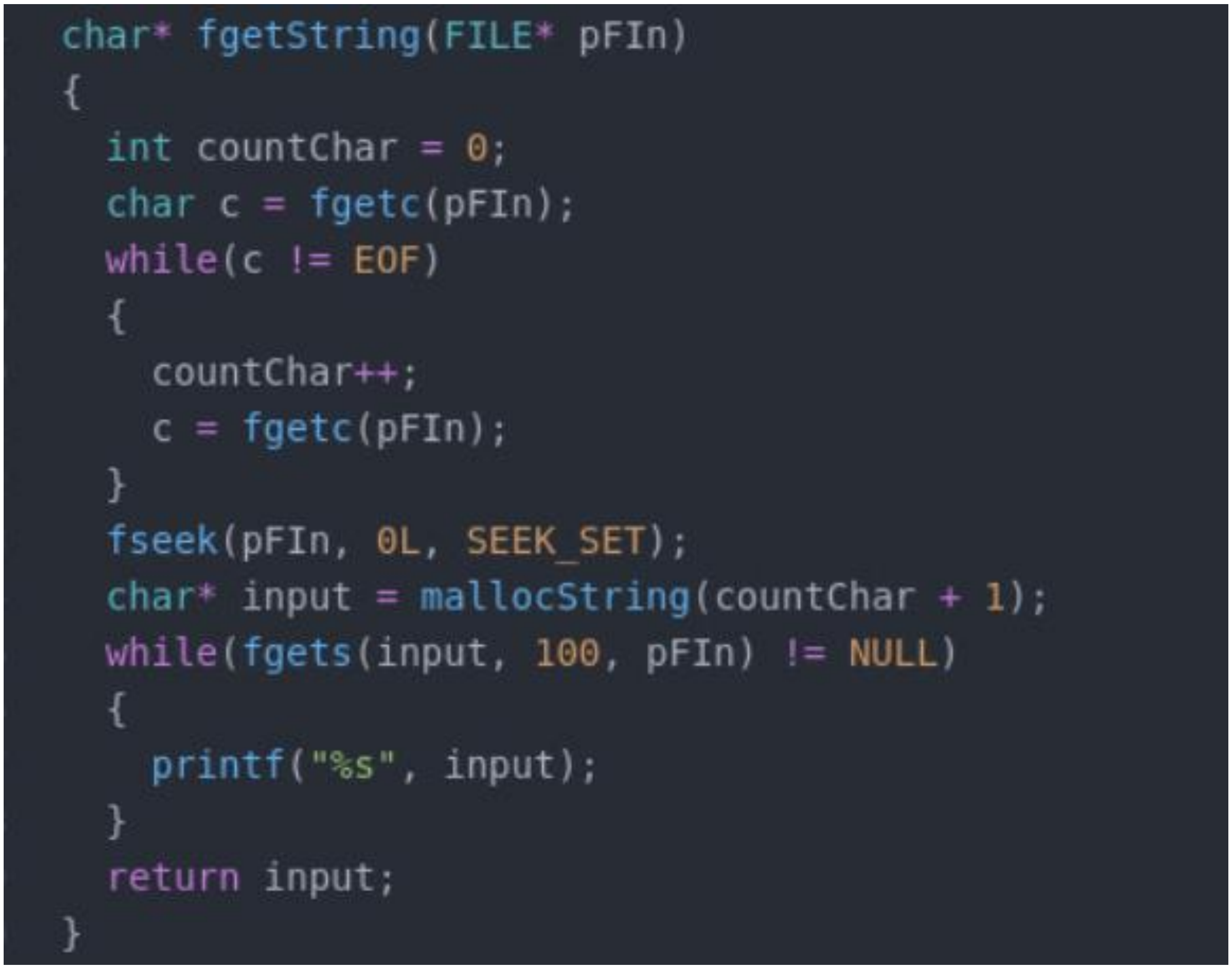
Figure 7: Content in Input1.txt

Exercise 2:

Question 1: Using your own functions from your Strings module to help, write a function

```
char* fgetString(FILE* pFIn);
```

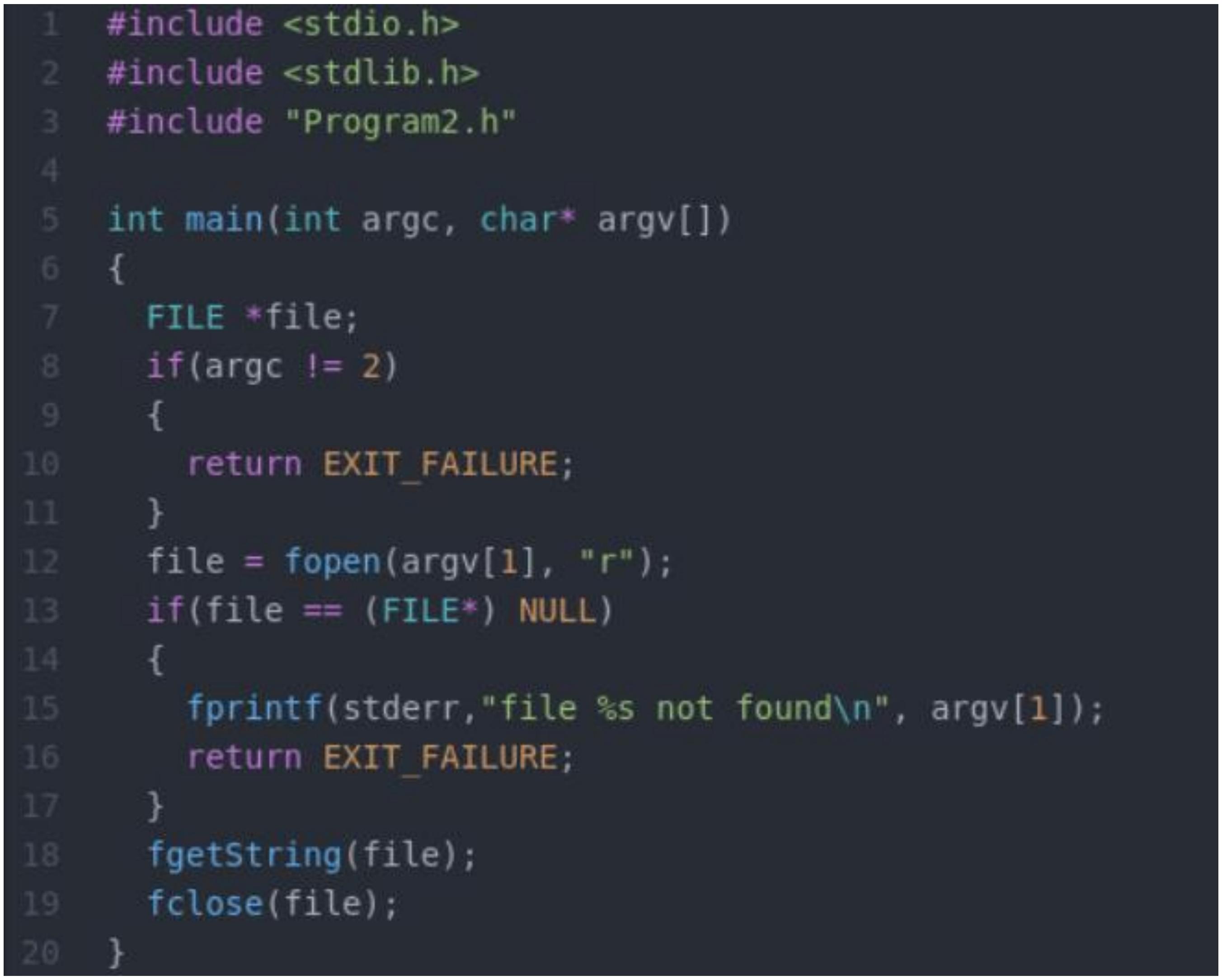
that reads the string length from an open file, allocates the memory, reads the string into it and returns it.



```
char* fgetString(FILE* pFIn)
{
  int countChar = 0;
  char c = fgetc(pFIn);
  while(c != EOF)
  {
    countChar++;
    c = fgetc(pFIn);
  }
  fseek(pFIn, 0L, SEEK_SET);
  char* input = mallocString(countChar + 1);
  while(fgets(input, 100, pFIn) != NULL)
  {
    printf("%s", input);
  }
  return input;
}
```

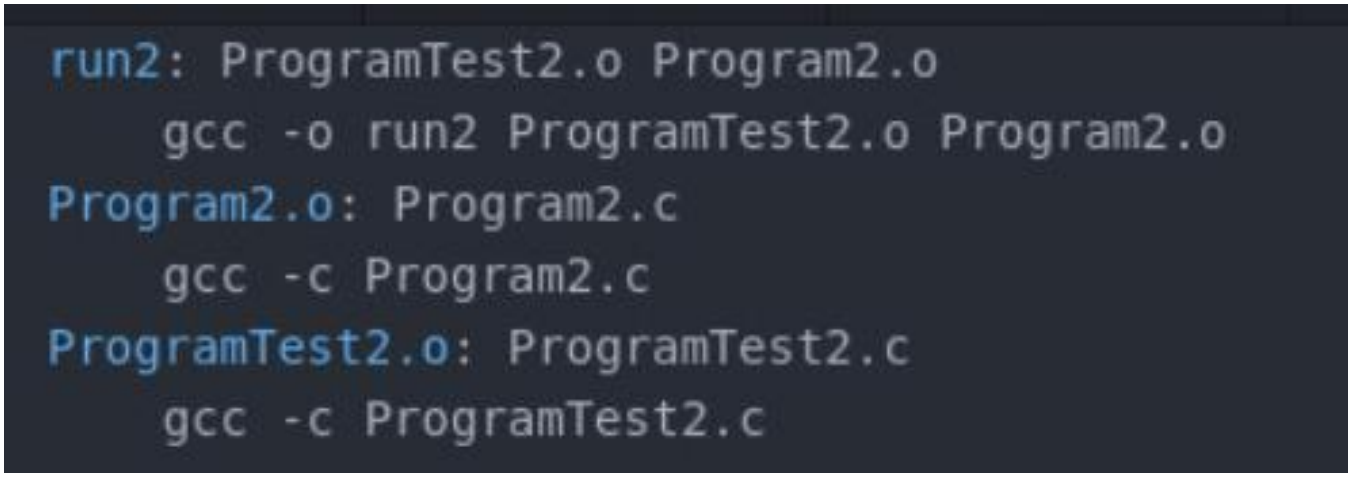
Figure 8: Source Code of fgetString()

Question 2: Now write a test driver program that opens a file (file name from the command line arguments) and reads all strings from a file and writes them, one string per line, to standard out. Of course you should add this as a new target in your Makefile before you get started



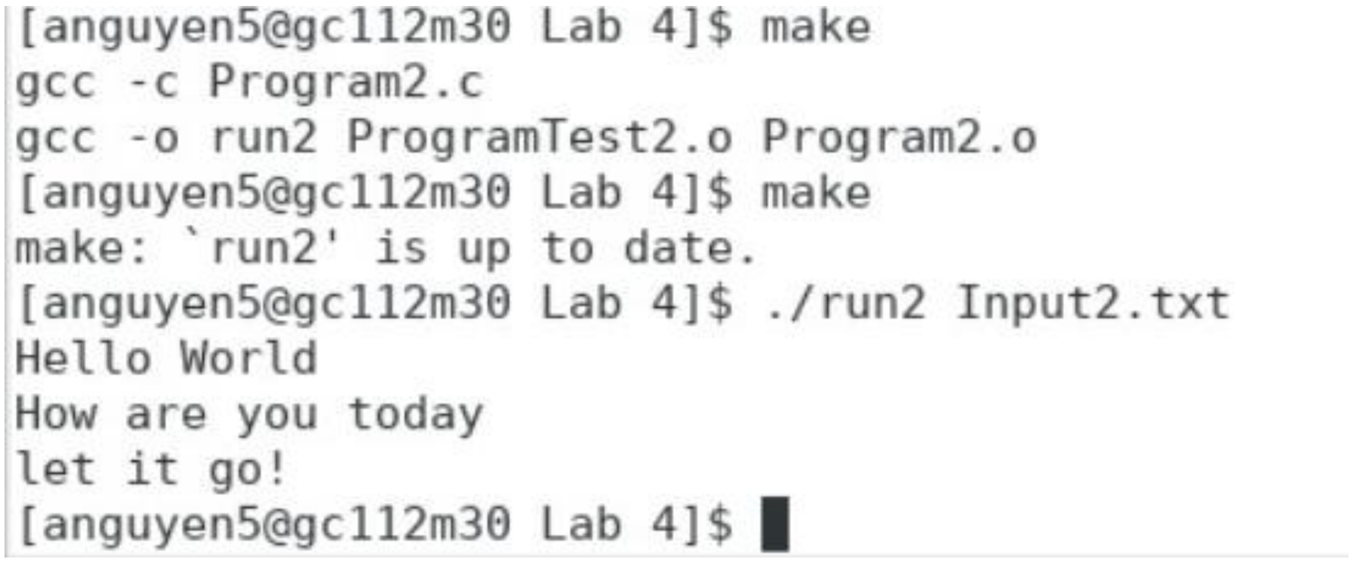
```c
#include <stdio.h>
#include <stdlib.h>
#include "Program2.h"

int main(int argc, char* argv[])
{
  FILE *file;
  if(argc != 2)
  {
    return EXIT_FAILURE;
  }
  file = fopen(argv[1], "r");
  if(file == (FILE*) NULL)
  {
    fprintf(stderr,"file %s not found\n", argv[1]);
    return EXIT_FAILURE;
  }
  fgetString(file);
  fclose(file);
}
```

Figure 9: Source Code of Test Driver Program



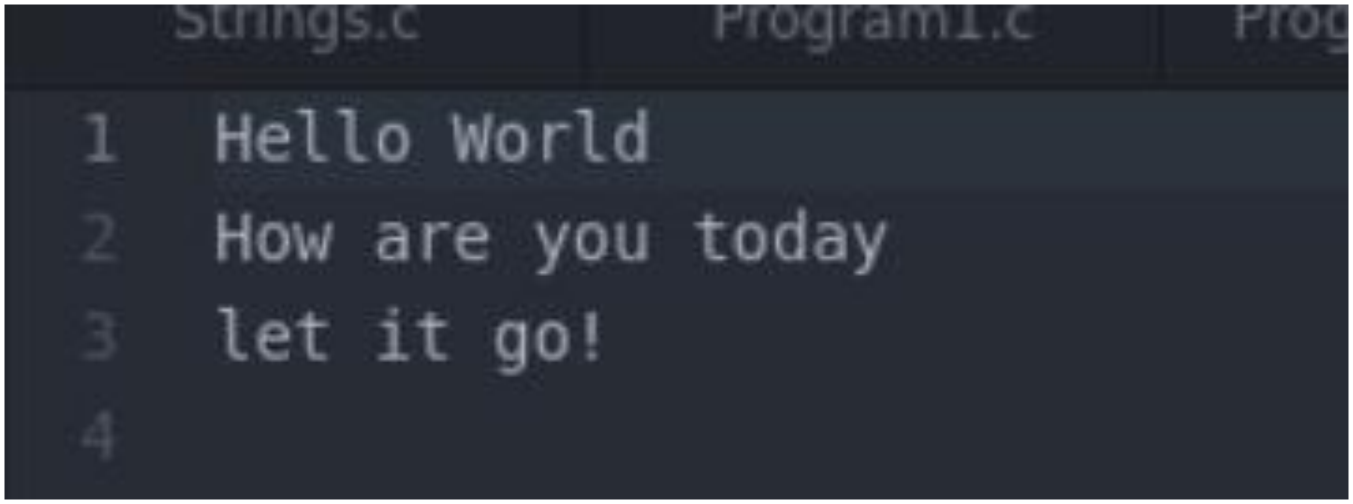
```makefile
run2: ProgramTest2.o Program2.o
    gcc -o run2 ProgramTest2.o Program2.o
Program2.o: Program2.c
    gcc -c Program2.c
ProgramTest2.o: ProgramTest2.c
    gcc -c ProgramTest2.c
```

Figure 10: makefile for Question 2



```
[anguyen5@gc112m30 Lab 4]$ make
gcc -c Program2.c
gcc -o run2 ProgramTest2.o Program2.o
[anguyen5@gc112m30 Lab 4]$ make
make: `run2' is up to date.
[anguyen5@gc112m30 Lab 4]$ ./run2 Input2.txt
Hello World
How are you today
let it go!
[anguyen5@gc112m30 Lab 4]$ 
```

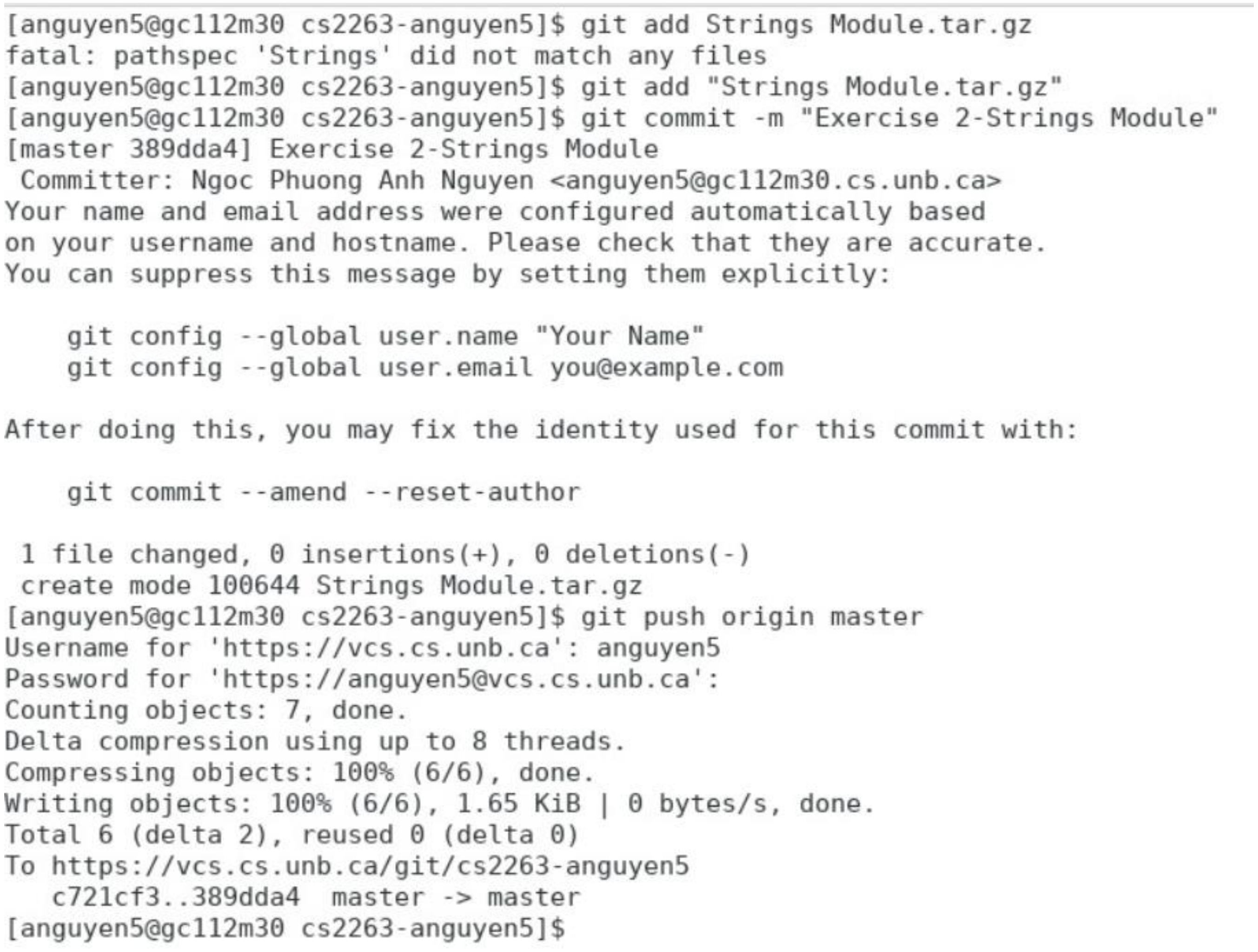
Figure 11: Output using makefile



```
Hello World
How are you today
let it go!

```

Figure 12: Content in Input2.txt

Exercise 3:



```
[anguyen5@gc112m30 cs2263-anguyen5]$ git add Strings Module.tar.gz
fatal: pathspec 'Strings' did not match any files
[anguyen5@gc112m30 cs2263-anguyen5]$ git add "Strings Module.tar.gz"
[anguyen5@gc112m30 cs2263-anguyen5]$ git commit -m "Exercise 2-Strings Module"
[master 389dda4] Exercise 2-Strings Module
 Committer: Ngoc Phuong Anh Nguyen <anguyen5@gc112m30.cs.unb.ca>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Strings Module.tar.gz
[anguyen5@gc112m30 cs2263-anguyen5]$ git push origin master
Username for 'https://vcs.cs.unb.ca': anguyen5
Password for 'https://anguyen5@vcs.cs.unb.ca':
Counting objects: 7, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.65 KiB | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
To https://vcs.cs.unb.ca/git/cs2263-anguyen5
   c721cf3..389dda4  master -> master
[anguyen5@gc112m30 cs2263-anguyen5]$
```