QUESTION:1

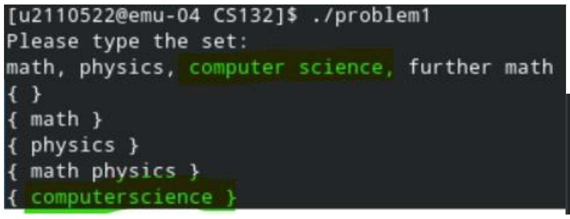
Powerset of set S is the set of all possible subsets of S.
In order to generate all possible elements of the powerset, I used binary numbers.
Suppose we have a set A of 4 elements: A = {a, b, c, d}. Then its powerset can be listed as follows:

0000	{}
0001	{d}
0010	{c}
0011	{c, d}
0100	{b}
0101	{b, d}
0110	{c, d}
0111	{b, c, d}
1000	{a}
1001	{a, d}
1010	{a, c}
1011	{a, c, d}
1100	{a, b}
1101	{a, b, d}
1110	{a, b, c}
1111	{a, b, c, d}

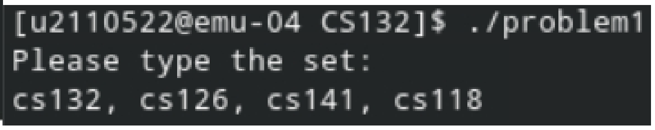
The above example shows one example of finding the elements of powerset of a set.
The number of bits depends on the number of elements in the original set.
In order to implement this idea, as an **input** we take set as a string, and **output** a string of set.
Input: string type
Output: string type
To move on with this implementation, we have to treat each element of the powerset as a single element. So we have divide the input string text into array words. In order to do this, I built a function **getWords()** which takes **string** as an **input** and returns **integer** which is the number of elements in that set. In addition to that it divides the **string** text into array **words[].**
The way it divides into words is based on the spaces and commas between elements of the set input. The format of the **string set input** must be as follows:

1. The input text must not be covered by any kind of braces;
2. The elements must be separated by using commas and spaces.
3. Between the characters of the elements, there must not be any space. If there is any then the function removes the spaces.

For example:



```
[u2110522@emu-04 CS132]$ ./problem1
Please type the set:
math, physics, computer science, further math
{ }
{ math }
{ physics }
{ math physics }
{ computerscience }
```

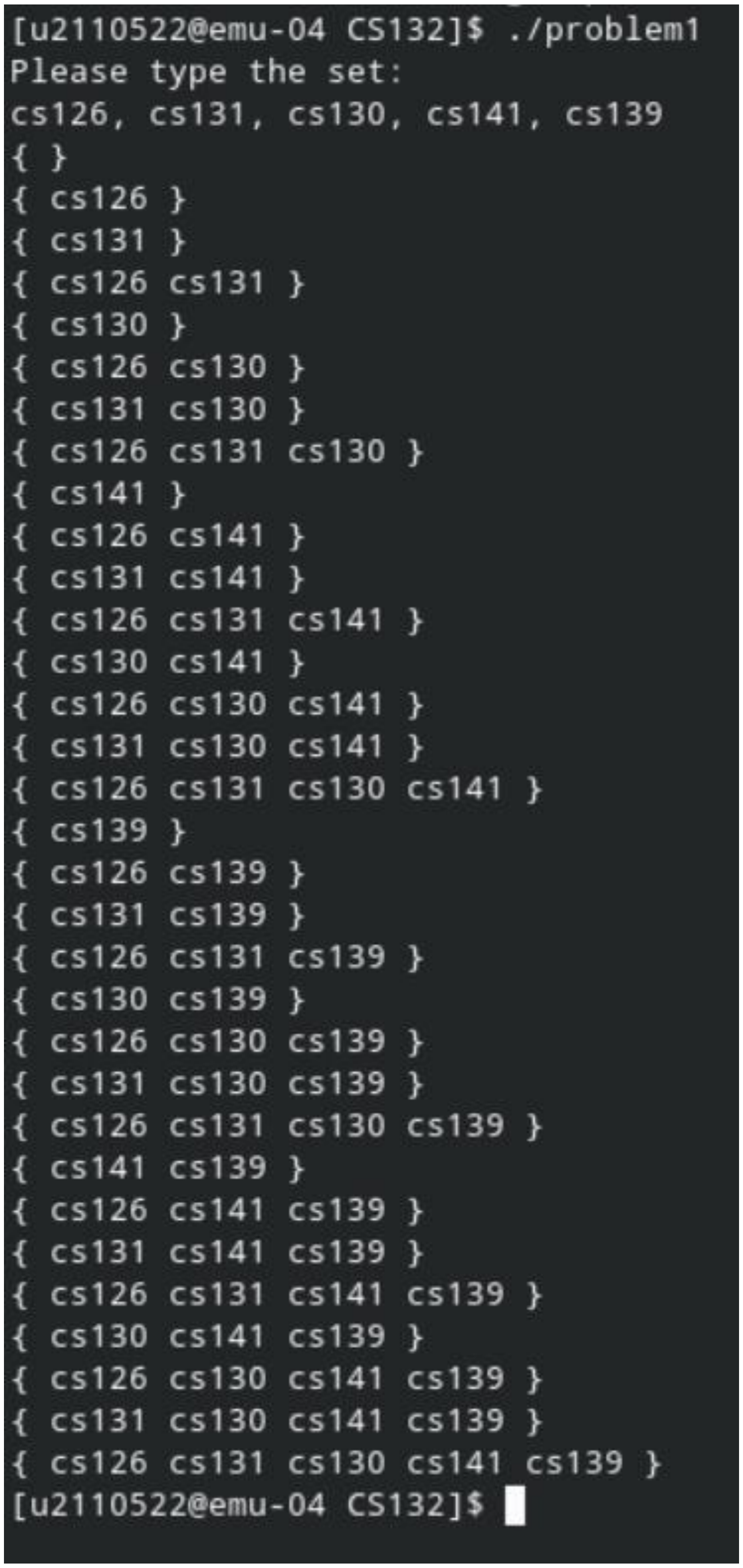


```
[u2110522@emu-04 CS132]$ ./problem1
Please type the set:
cs132, cs126, cs141, cs118
```

We can implement binary system using arrays where each digit of the binary number is stored in arrays. As it can be seen from the above table, on each run the value of the binary number is incremented. In order to apply this characteristics I used (void) ***increment()*** function which takes array of binary numbers and number of words as input and does not return anything.

So in the main method, I looped through 2 pow n times. In each cycle, by using number of 1s and their positions I outputted all the possible subsets of a set following that used increment() method to increment the value of the binary number.

So the time complexity of the method I used was O(2^n) (n – input size in this case |S|) which is not considered as efficient.



```
[u2110522@emu-04 CS132]$ ./problem1
Please type the set:
cs126, cs131, cs130, cs141, cs139
{ }
{ cs126 }
{ cs131 }
{ cs126 cs131 }
{ cs130 }
{ cs126 cs130 }
{ cs131 cs130 }
{ cs126 cs131 cs130 }
{ cs141 }
{ cs126 cs141 }
{ cs131 cs141 }
{ cs126 cs131 cs141 }
{ cs130 cs141 }
{ cs126 cs130 cs141 }
{ cs131 cs130 cs141 }
{ cs126 cs131 cs130 cs141 }
{ cs139 }
{ cs126 cs139 }
{ cs131 cs139 }
{ cs126 cs131 cs139 }
{ cs130 cs139 }
{ cs126 cs130 cs139 }
{ cs131 cs130 cs139 }
{ cs126 cs131 cs130 cs139 }
{ cs141 cs139 }
{ cs126 cs141 cs139 }
{ cs131 cs141 cs139 }
{ cs126 cs131 cs141 cs139 }
{ cs130 cs141 cs139 }
{ cs126 cs130 cs141 cs139 }
{ cs131 cs130 cs141 cs139 }
{ cs126 cs131 cs130 cs141 cs139 }
[u2110522@emu-04 CS132]$
```

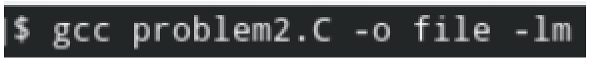
Figure 1 Example of outputs. The elements in the output has braces around and elements are separated by space.

QUESTION 2

Running the code:

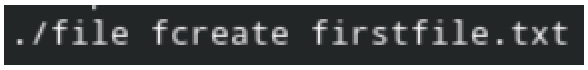
In order to run the code, the program must be in same directory as the users.

In order to compile, the following code must be written in terminal:



```
$ gcc problem2.C -o file -lm
```

In order to run and use the file handling functions, the user has to use the following commands (example):

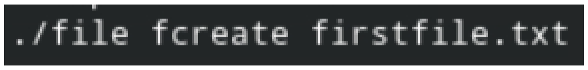


```
./file fcreate firstfile.txt
```

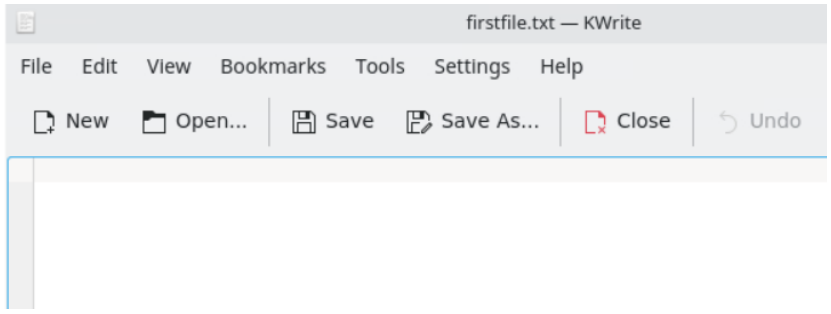
The above command is used to create (using fcreate) textfile firstfile.txt. Every command must start with **./file.**

File Operations:

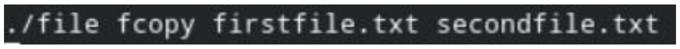
fcreate <filename.txt> - used to create a file with name *filename.* This is one of the simplest built command.
The bellow command created an empty file - firstfile.txt.



```
./file fcreate firstfile.txt
```

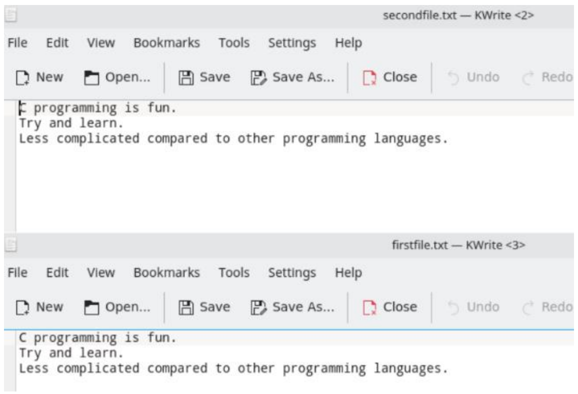

firstfile.txt — KWrite
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo

fcopy <fromfile.txt> <tofile.txt> - copy the contents of the file *fromfile.txt* to *tofile.txt.*



```
./file fcopy firstfile.txt secondfile.txt
```

So the command to the left copies the contents of the firstfile.txt to secondfile.txt. It is not necessary to write separate command to create secondfile.txt.


secondfile.txt — KWrite <2>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo Redo

```
C programming is fun.
Try and learn.
Less complicated compared to other programming languages.
```

firstfile.txt — KWrite <3>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo Redo

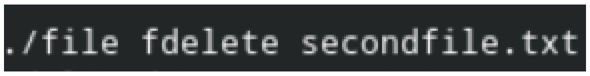
```
C programming is fun.
Try and learn.
Less complicated compared to other programming languages.
```

The second file is created automatically when we wrote:

fp = fopen (secondfile, “w”);

If there is no such file, mode “w” allows to automatically create and copy the contents of the firstfile.txt.

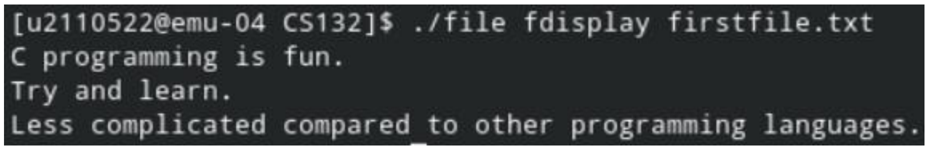
fdelete <filename.txt> - delete a file with name *filename.*



```
./file fdelete secondfile.txt
```

The command is made possible with ***remove()*** function. If it is successful then it gives a message. Otherwise it throws an error.

fdisplay <filename.txt> - display the contents of an existing file with a name *filename.txt.*

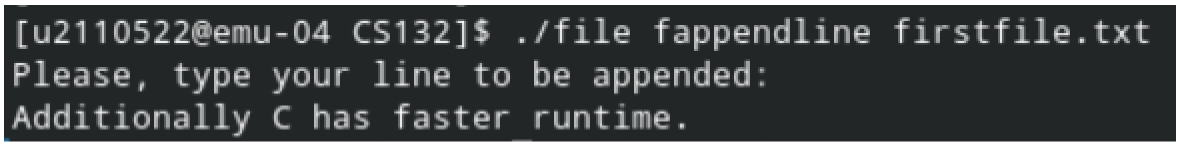


```
[u2110522@emu-04 CS132]$ ./file fdisplay firstfile.txt
C programming is fun.
Try and learn.
Less complicated compared to other programming languages.
```

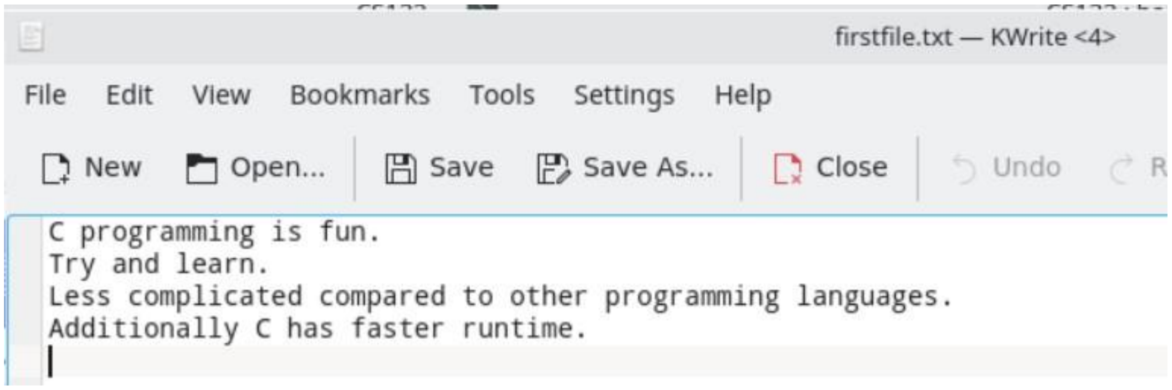
.

Line Operations

fappendline <filename.txt> – create a new line of *content* at the end of file with name *filename.txt.* Ask for the content after getting specified the name of the file.



```
[u2110522@emu-04 CS132]$ ./file fappendline firstfile.txt
Please, type your line to be appended:
Additionally C has faster runtime.
```

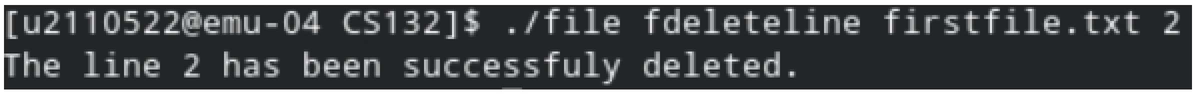


firstfile.txt — KWrite <4>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo

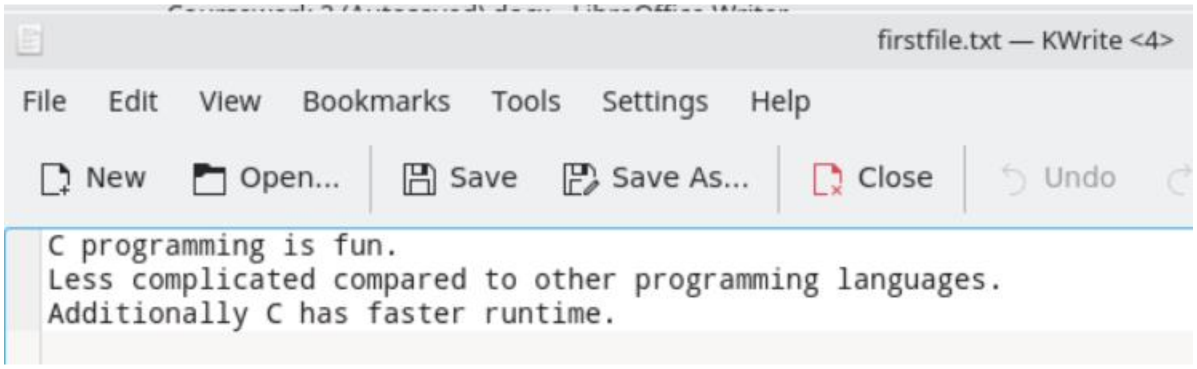
```
C programming is fun.
Try and learn.
Less complicated compared to other programming languages.
Additionally C has faster runtime.
```

In order to write we used: fp = fopen(firstfile, “a”); (in ‘append’ mode).

fdeleteline <filename.txt> {line number} – delete a line of content at a particular *line number* in a file with name *filename.txt.*



```
[u2110522@emu-04 CS132]$ ./file fdeleteline firstfile.txt 2
The line 2 has been successfuly deleted.
```



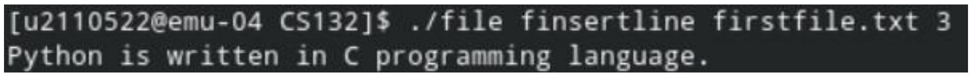
firstfile.txt — KWrite <4>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo

```
C programming is fun.
Less complicated compared to other programming languages.
Additionally C has faster runtime.
```

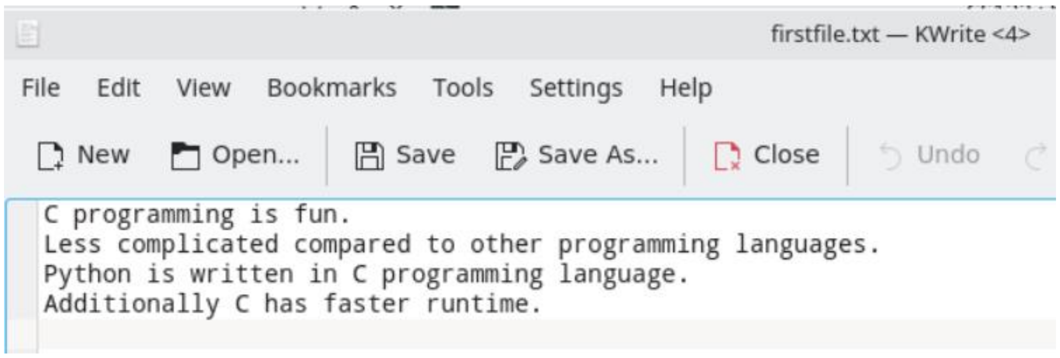
To implement this functionality, I first copied the contents of the firstfile.txt, except for the content in the 2nd line, to the temporarily created textfile. Then the contents of the temporary file is copied back to the firstfile.txt. The temporary file has been deleted.

finsertline <filename.txt> {line number} – create a new line of content at a particular *line number* in a file with name *filename.txt.*

Ask for the content after getting specified the name of the file.



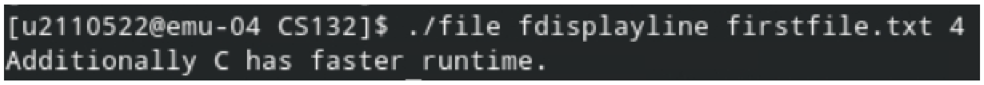
```
[u2110522@emu-04 CS132]$ ./file finsertline firstfile.txt 3
Python is written in C programming language.
```


firstfile.txt — KWrite <4>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo

```
C programming is fun.
Less complicated compared to other programming languages.
Python is written in C programming language.
Additionally C has faster runtime.
```

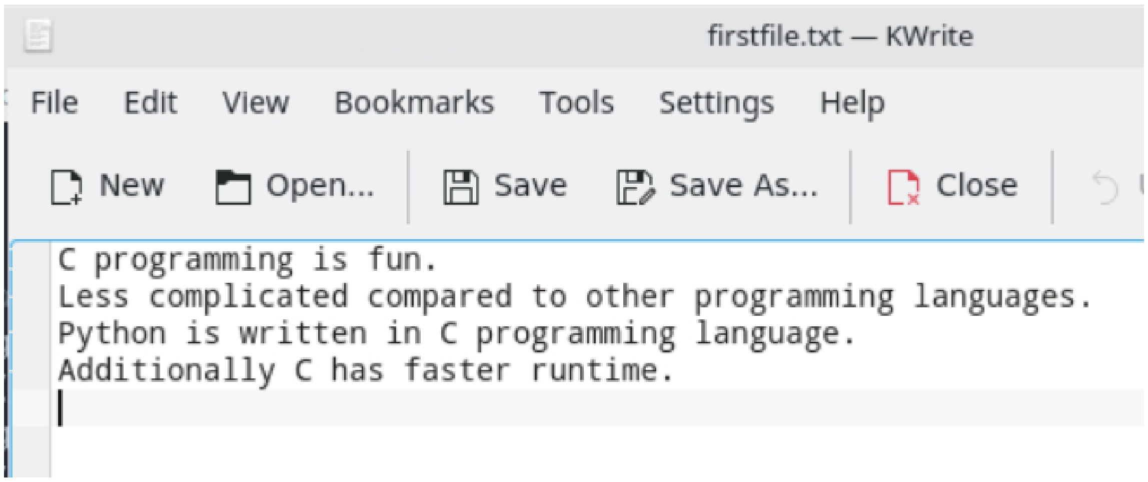
In order to implement this functionality, I implemented a technique similar to the previous ones. I have created temporary file and copied the contents of the original file to the temporary one including the new line input by the user. Then I copied back the contents in temporary file to the original firstfile.txt file.

fdisplayline <filename.txt> {line number} – display the contents of a file at a particular *line number* in a file with name *filename.txt.*



```
[u2110522@emu-04 CS132]$ ./file fdisplayline firstfile.txt 4
Additionally C has faster runtime.
```

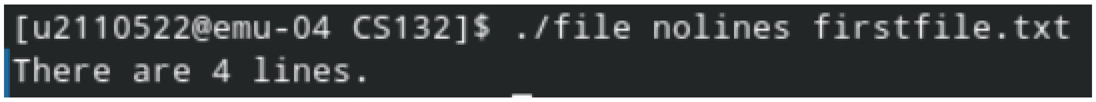
As it can be seen above, the 4th line of the text file firstfile.txt was called and the result was the same as the 4th line in the picture below.


firstfile.txt — KWrite
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close

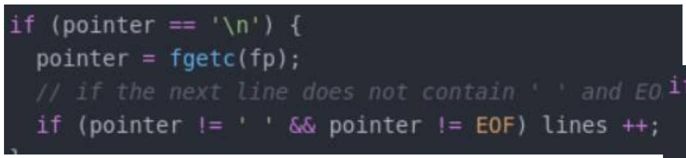
```
C programming is fun.
Less complicated compared to other programming languages.
Python is written in C programming language.
Additionally C has faster runtime.
```

In order to implement this command, I just traced the lines that algorithm points. Once I reached the specified line, I used printf() method to display it.

nolines <filename.txt> – show the number of lines in a file with name *filename.txt*.

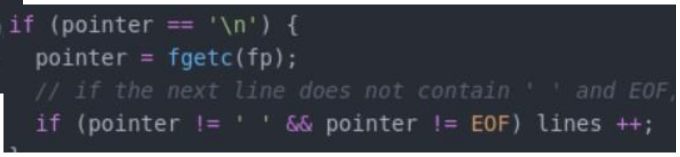


```
[u2110522@emu-04 CS132]$ ./file nolines firstfile.txt
There are 4 lines.
```



```
if (pointer == '\n') {
  pointer = fgetc(fp);
  // if the next line does not contain ' ' and EO
  if (pointer != ' ' && pointer != EOF) lines ++;
```

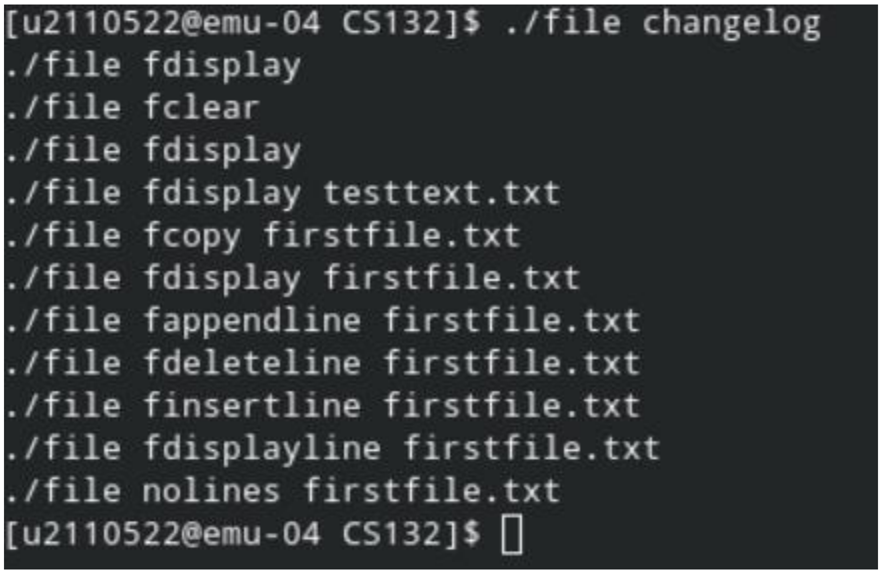
The number of lines of text is found by counting the number of '\n' lines. In some cases this might not be the case.



```
if (pointer == '\n') {
  pointer = fgetc(fp);
  // if the next line does not contain ' ' and EOF,
  if (pointer != ' ' && pointer != EOF) lines ++;
```

Therefore I included one more condition into my code:

changelog - Display the sequence of operations performed on all files created by your program, including the number of lines following each operation.



```
[u2110522@emu-04 CS132]$ ./file changelog
./file fdisplay
./file fclear
./file fdisplay
./file fdisplay testtext.txt
./file fcopy firstfile.txt
./file fdisplay firstfile.txt
./file fappendline firstfile.txt
./file fdeleteline firstfile.txt
./file finsertline firstfile.txt
./file fdisplayline firstfile.txt
./file nolines firstfile.txt
[u2110522@emu-04 CS132]$ 
```

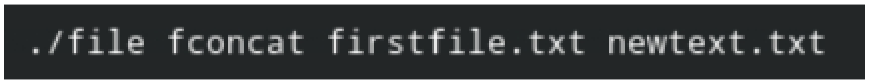
Before doing any operation, I decided to create text file in which all the change log operations on any text files are stored.

After doing the operations on text files, these operations are put into change log file.

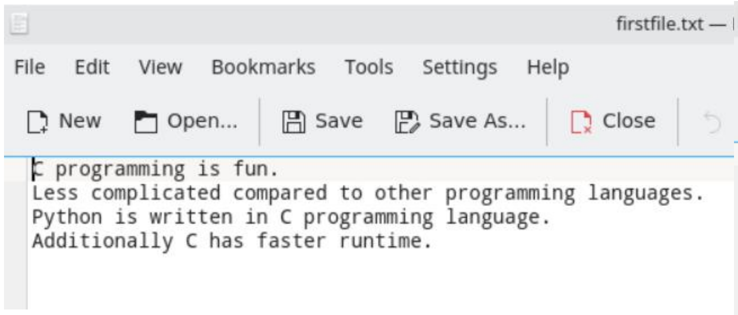
And when requested they are displayed in console.

fconcat <fromfile.txt> <tofile.txt> - concatenates the contents of the fromfile.txt to tofile.txt.

From my research concatenating the contents of files (such as datasets and etc.) has a lot applications in collecting data used for training ML models.

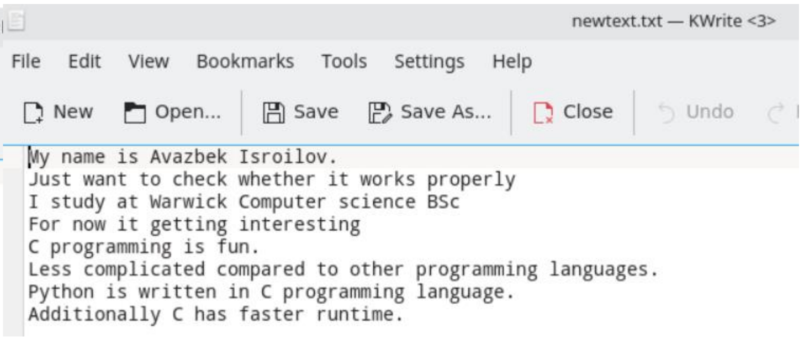


```
./file fconcat firstfile.txt newtext.txt
```



firstfile.txt —
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close

```
C programming is fun.
Less complicated compared to other programming languages.
Python is written in C programming language.
Additionally C has faster runtime.
```

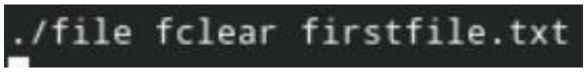


newtext.txt — KWrite <3>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo

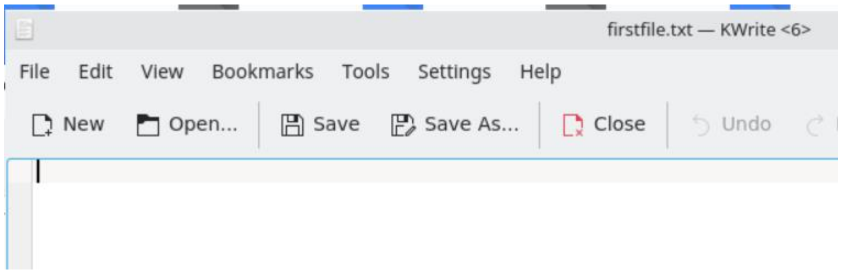
```
My name is Avazbek Isroilov.
Just want to check whether it works properly
I study at Warwick Computer science BSc
For now it getting interesting
C programming is fun.
Less complicated compared to other programming languages.
Python is written in C programming language.
Additionally C has faster runtime.
```

fclear <textfile.txt> - clears the contents of the file.

When I was testing my code in this coursework, I had to clean many of the files to continue with my testing. Even though it is still possible to implement fclear using other basic operator first fdelete, followed by fcreate, fclear is much more convenient.



```
./file fclear firstfile.txt
```


firstfile.txt — KWrite <6>
File Edit View Bookmarks Tools Settings Help
New Open... Save Save As... Close Undo