FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)

Compiler Construction - COMP 451 A

Spring 22

Lab 03

Muhammad Sulaiman Sultan - 231453415

Muhammad Sameed Gilani - 231488347

INTRODUCTION:

- **Command line arguments:**
  - ➔ int argc:

    This stores the number of arguments passed by the user when the run the program. This also includes the program name.

  - ➔ Char* argv[]:

    This is an array of character pointers, which point to the address of the command line arguments that were passed. argv[0] always stores the program name.

- **fopen() → <stdio.h>**
  - ■ This is a C library function that uses a file pointer to open a file with in the chosen mode. In our case, “r” (read) was used.
  - ■ It returns a file pointer if file opens successfully or Null when it fails.
  - ■ We use this function to open and read the text file.

Function Signature:

```
FILE *fopen(const char *file_name, const char
*mode_of_operation);
```

Sample from program:

```
FILE *fp;

fp = fopen(argv[1],"r");


if (fp == NULL){

     printf("Error opening file");
```

```
        exit(0);
    }
```

- **malloc() → <stdlib.h>**
    - This function is used to dynamically allocate a single block of memory in the heap, with a size of our choice.
    - It returns a pointer to the allocated memory.
    - It does not initialize memory on execution.
    - We use this to allocate a block of memory to store the each character read from the file

  Function Signature:

```
ptr = (cast-type*) malloc(byte-size)
```

  Sample from program:

```
char* buff = (char*) malloc(sizeof(char)*1000);
```

    - sizeof() returns the size of the data type in bytes. (2 for char). So this allocates a memory of 2000 bytes. With buff as the pointer.

- **fgetc() → <stdio.h>**
    - It takes a file pointer as input and returns a single char read from the file.
    - It returns type int however, as character literal has type int.
    - We use this function to read a file, character by character

  Function Signature:

```
int fgetc(FILE *pointer)
```

Sample From Prorgram:

```
c = fgetc(fp);
```

c is then stored in the memory, previously allocated.

- **fputs() → <stdio.h>**
  - fputs() allows us to write text to a desired file.
  - In our case we use it to print text on the screen, using stdout

Function Signature:

```
int fputs (const char * str,FILE*stream);
```

Sample from program:

```
fputs(buff, stdout);
```

- **rewind() → <stdio.h>**
  - rewind is used to set the file pointer back to the beginning of the file.
  - We use rewind in the program to read the file twice, after reading it to the end once.

Function Signature:

```
void rewind(FILE *stream)
```

Sample From Program:

```
rewind(fp);
```

- **RemoveBlankLines() → User Defined**
  - It takes a file name as input
  - It opens and reads the file given. It then removes all blank(empty) lines in the file and then prints the text read from the file.

Function Signature:

```
void removeBlankLines(char *);
```

From the program:

```
void removeBlankLines(char *str)

{

FILE *fp;

fp = fopen(str,"r");

if(fp == NULL)

{

     printf("Error opening file.");

     exit(0);

}

char* buff = (char*) malloc(sizeof(char)*1000);

char c = ' ';

int len = 0;


while(c != EOF)

{

     c = fgetc(fp);

     printf("%c",c);

}

printf("\n");

c = ' ';

rewind(fp);

while(c != EOF)
```

```
        {
                c = fgetc(fp);
                if (buff[len-1] == '\n' && c == '\n')
                        continue;
                buff[len] = c;
                len++;
        }
        fputs(buff, stdout);
}
```

LOGIC:

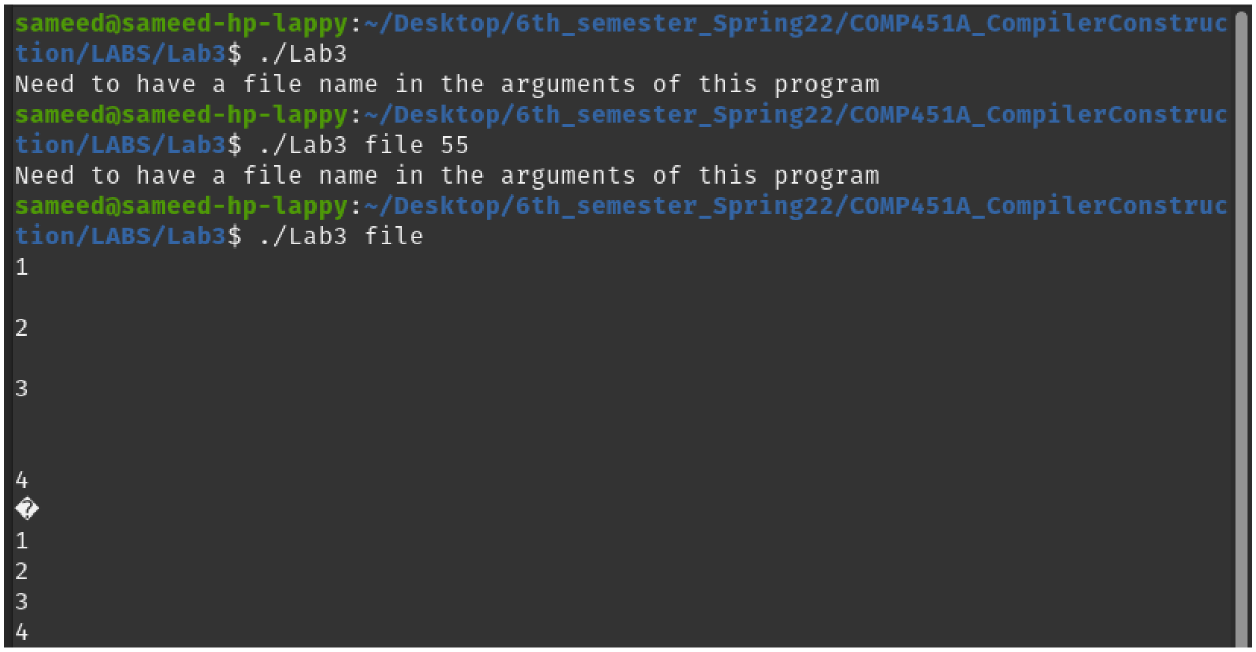
- In main we have a check for the number of command line arguments given. If they are less then 2 (the c file name and the text file), the program will exit.
- We give the text file as argument to, RemoveBlankLines().
- In RemoveBlankLines(), we open the file to be read. Program exits if the file isnt opened.
- “int len” is used to index the buffer. “char c” is used to to store the character read by fgetc().
- We create dynamic memory for 2000 bytes with malloc(). In this bufffer we will store the characters read in the file.
- A while loop runs while the end of file isnt reached. In this loop we read the file character by character with fgetc(), and we print the file as it was given. (With blank spaces)
- We use rewind(), to bring the pointer back to the start of the file
- Another while loop runs until end of file is reached. In this loop we read the file character by charcter and store it in buff and increment “int len”
  But if a newline character is read, we do not add it to the buffer and continue.
- Once the loop ends we print out, buff with stdout.

CODE

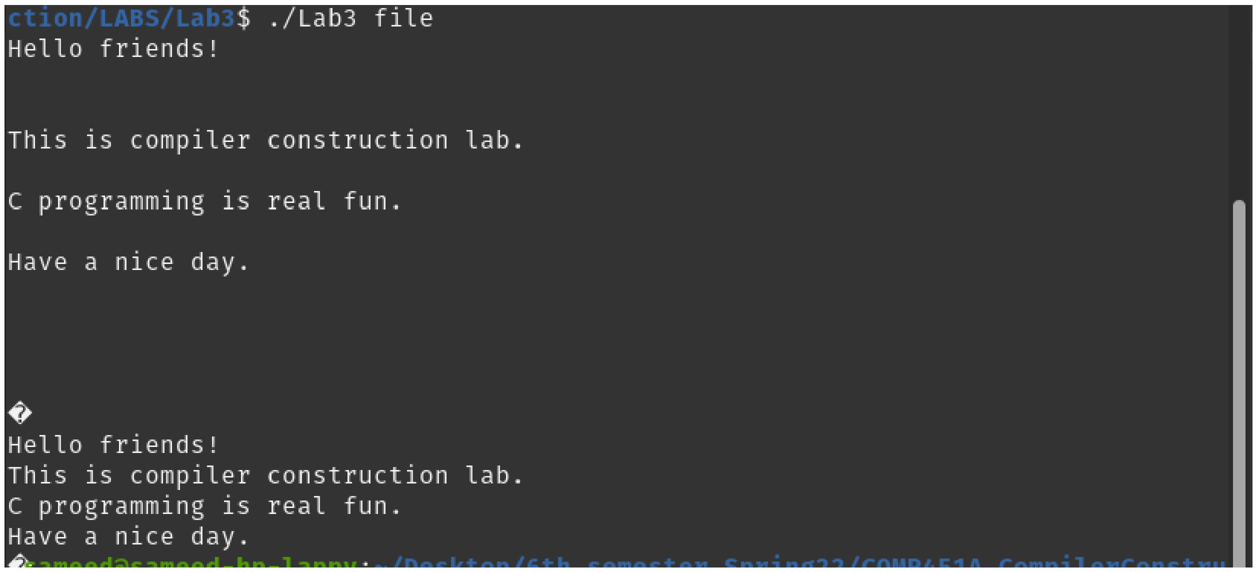
```
#include <stdio.h>

#include <stdlib.h>


void removeBlankLines(char *);


int main(int argc, char *argv[])

{

	if(argc != 2)

	{

		printf("Need to have a file name in the arguments of this 
program\n");

		exit(0);

	}


	removeBlankLines(argv[1]);

	return 0;

}


void removeBlankLines(char *str)

{

	FILE *fp;

	fp = fopen(str,"r");

	if(fp == NULL)

	{

		printf("Error opening file.");

		exit(0);

	}

	char* buff = (char*) malloc(sizeof(char)*1000);
```

```
        char c = ' ';
        int len = 0;

        while(c != EOF)
        {
              c = fgetc(fp);
              printf("%c",c);
        }
        printf("\n");
        c = ' ';
        rewind(fp);
        while(c != EOF)
        {
              c = fgetc(fp);
              if (buff[len-1] == '\n' && c == '\n')
                    continue;
              buff[len] = c;
              len++;
        }
        fputs(buff, stdout);
}
```

Sample Outputs:



```
sameed@sameed-hp-lappy:~/Desktop/6th_semester_Spring22/COMP451A_CompilerConstruc
tion/LABS/Lab3$ ./Lab3
Need to have a file name in the arguments of this program
sameed@sameed-hp-lappy:~/Desktop/6th_semester_Spring22/COMP451A_CompilerConstruc
tion/LABS/Lab3$ ./Lab3 file 55
Need to have a file name in the arguments of this program
sameed@sameed-hp-lappy:~/Desktop/6th_semester_Spring22/COMP451A_CompilerConstruc
tion/LABS/Lab3$ ./Lab3 file
1

2

3


4
�
1
2
3
4
```



```
ction/LABS/Lab3$ ./Lab3 file
Hello friends!


This is compiler construction lab.

C programming is real fun.

Have a nice day.




�
Hello friends!
This is compiler construction lab.
C programming is real fun.
Have a nice day.
�sameed@sameed-hp-lappy:~/Desktop/6th_semester_Spring22/COMP451A_CompilerConstru
```