Lab 01: Scanning and filtering the source program comments

Write a program which will take a C source program as input, consisting of single and multi-line comments and multiple blank spaces and produces another source file removing the comments and single blank space. Display that new source file produced as the output of the program.

// This is a single line comment

/* This is a

Multiline Comment */

Useful functions needed for the task

getc(), fgetc(), getchar()

Get a single character from the console or from a file.

Prototypes

```
#include <stdio.h>

int getc(FILE *stream);
int fgetc(FILE *stream);
int getchar(void);
```

Description

All of these functions in one way or another , read a single character from the console or from a `FILE`. The differences are fairly minor, and here are the descriptions:

`getc()or fgetc()` returns a character from the specified `FILE`. From a usage standpoint, **`getc()`** equivalent to the same **`fgetc()`** call, and **`fgetc()`** is a little more common to see. Only the implementation of the two functions differs.

`getchar()` returns a character from *`stdin`*. In fact, it's the same as calling `getc(stdin)`.

Return Value

All three functions return the `unsigned char` that they read, except it's cast to an `int`.

If end-of-file or an error is encountered, all three functions return `EOF`.

Example

```
// read all characters from a file, outputting only the letter 'b's
// it finds in the file

#include <stdio.h>

int main(void)
{
    FILE *fp;
    int c;

    fp = fopen("datafile.txt", "r");

    // this while-statement assigns into c, and then checks against EOF:

    while((c = fgetc(fp)) != EOF) {
        if (c == 'b') {
            printf("%c",c);
        }
    }

    fclose(fp);

    return 0;
}
```

putc(), fputc(), putchar()

Write a single character to the console or to a file.

Prototypes

```
#include <stdio.h>

int putc(int c, FILE *stream);
int fputc(int c, FILE *stream);
int putchar(int c);
```

Description

All three functions output a single character, either to the console or to a `FILE`.

`putc()` takes a character argument, and outputs it to the specified `FILE`. **`fputc()`** does exactly the same thing, and differs from **`putc()`** in implementation only. Most people use **`fputc()`**.

`putchar()` writes the character to the console, and is the same as calling `putc(c, stdout)`.

Return Value

All three functions return the character written on success, or `EOF` on error.

Example

```
#include <stdio.h>

int main ()
{
   FILE *fp;
   int ch;

   fp = fopen("file.txt", "w");
   for( ch = 33 ; ch <= 100; ch++ )
   {
      putc(ch, fp);
   }
   fclose(fp);

   return(0);
}
```