CS2211a Lab No. 11
Introduction to C
Take home lab

The objective of this lab is to practice:

- Input/output functions

==

1. The following function is supposed to print the number of periods in a file. Unfortunately, it did not print the correct number of periods. Find the error in the function and show how to fix it. Use the following Unix command to genetare your test file

```
cat <<+ > file.txt
abc.....def
+

#include <stdio.h>

int count_periods(const char *filename)
{
  FILE *fp;
  int n = 0;
  if((fp = fopen(filename, "r")) != NULL)
  { while(fgetc(fp) != EOF)
      if(fgetc(fp) == '.')
        n++;
    fclose (fp);
  }
  return n;
}

int main (void)
{
    printf("%d\n", count_periods("file.txt"));
    return 0;
}
```

2. Predict the output of each of the following function calls and then write a program to verify your answers.

```
printf("%c", '\n');
printf("%s", "\n");
printf("\n");
putchar('\n');
puts("\n");
```

3. Write a program to practice the differences between the following function calls.

```
fopen(file_name, "r");
fopen(file_name, "w");
fopen(file_name, "a");
fopen(file_name, "r+");
fopen(file_name, "w+");
fopen(file_name, "a+");
```

4. The following program generates a binary data file which contains 100 structures. Read the following program carefully, compile it, and run it. Now, modify this program by writing another function that will reopen this data file again to read and print the first, the fifth, second last and the last structures in the file. *Restriction: you are not allowed to upload the entire data file to an array.* Yet, you are only allowed to use `fseek` to directly access the required structions. Test your function by calling it from the main function.

```
# include <stdio.h>

struct data
{ int   value_1;
  float value_2;
};

int create_data(const char *filename)
{ FILE *fp;
  struct data my_data[100];
  int i, n = 0;

  for(i=0; i<100; i++)
  { my_data[i].value_1 = i;
    (my_data + i)->value_2 = i;
  }

  if((fp = fopen(filename, "wb")) != NULL)
  { n = fwrite(my_data, sizeof(struct data),100, fp);
    fclose (fp);
  }
  return n;
}

int main (void)
{
  printf("%d\n", create_data("file.binary"));
  return 0;
}
```

5. Write a program which assigns the value 1234567.1234567 to a float variable. Store the value of this variable to a file using `fprintf` function call. Store the value again of this variable to another file using `fwrite` function call. Compare the size of the two files. Try to print the content of the two files using `cat` Unix command. Compare the outputs.

6. Write a program named `fcat` that "concatenates" any number of files by writing them to standard output, one after the other, with no break between files. For example, the following command will display the files `f1.c, f2.c,` and `f3.c` on the screen:
`fcat f1.c f2.c f3.c`
`fcat` should issue an error message if any file can't be opened. *Hint:* Since it has no more than one file open at a time, `fcat` needs only a single file pointer variable. Once it's finished with a file, `fcat` can use the same variable when it opens the next file.

Modify the program to store the output in a file provided by the user as the last argument in the command line.