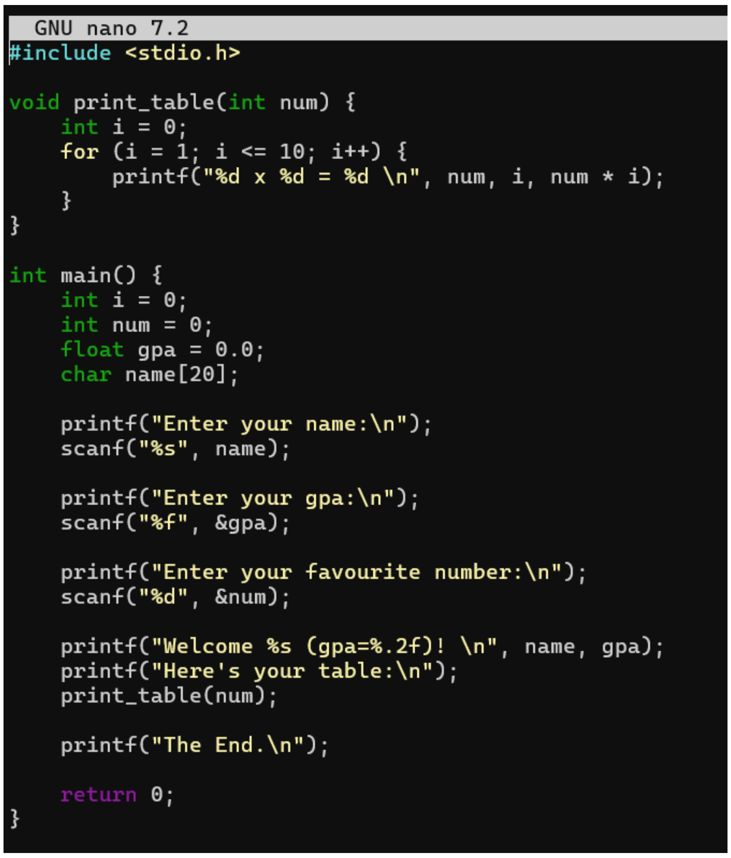
Lab 4 OS

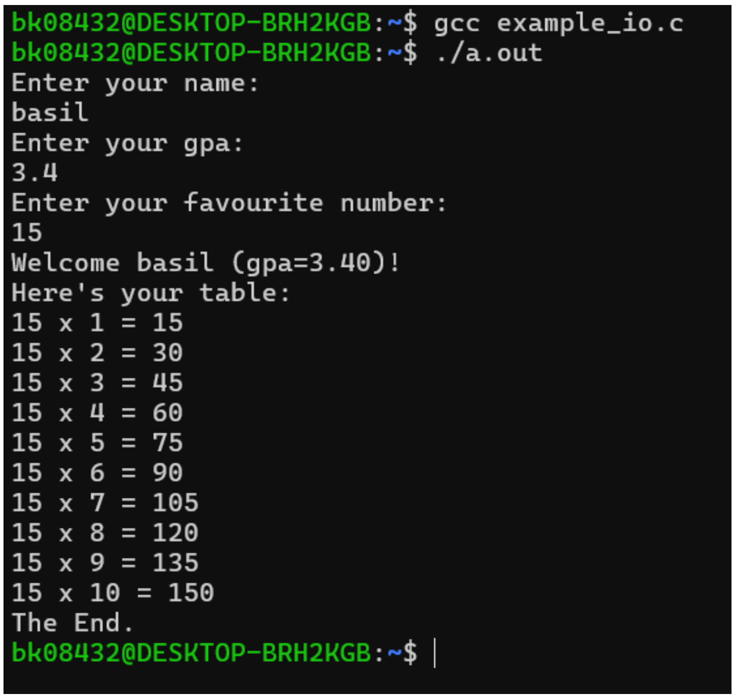
Name: basil khowaja bk08432

Q1)

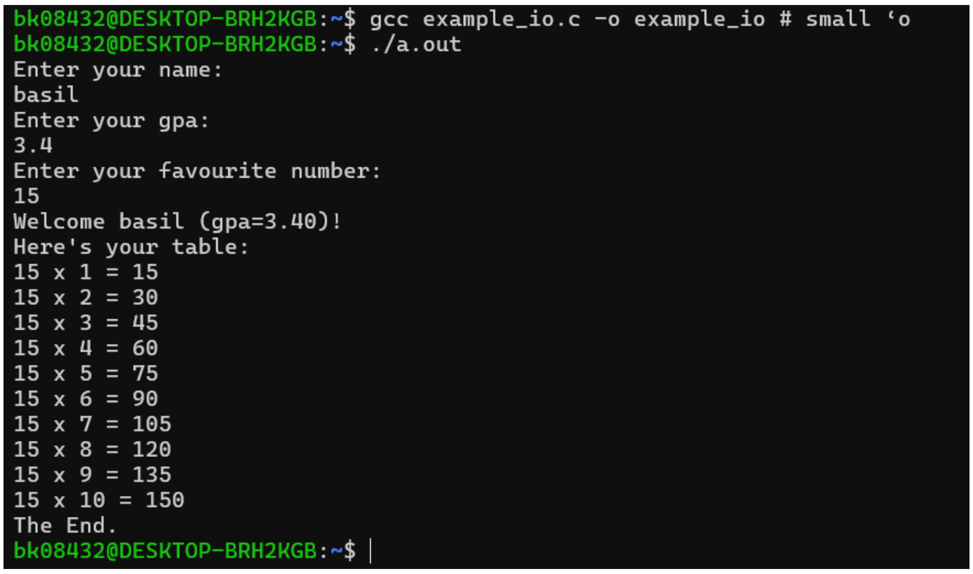
Code:



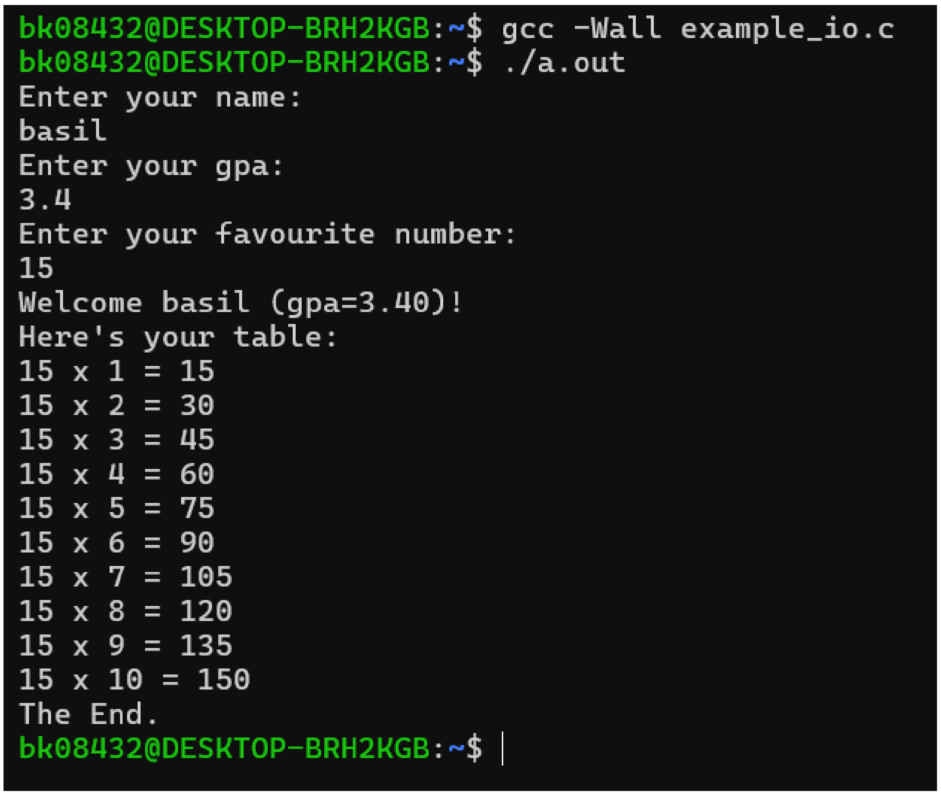
```
  GNU nano 7.2
#include <stdio.h>

void print_table(int num) {
    int i = 0;
    for (i = 1; i <= 10; i++) {
        printf("%d x %d = %d \n", num, i, num * i);
    }
}

int main() {
    int i = 0;
    int num = 0;
    float gpa = 0.0;
    char name[20];

    printf("Enter your name:\n");
    scanf("%s", name);

    printf("Enter your gpa:\n");
    scanf("%f", &gpa);

    printf("Enter your favourite number:\n");
    scanf("%d", &num);

    printf("Welcome %s (gpa=%.2f)! \n", name, gpa);
    printf("Here's your table:\n");
    print_table(num);

    printf("The End.\n");

    return 0;
}
```



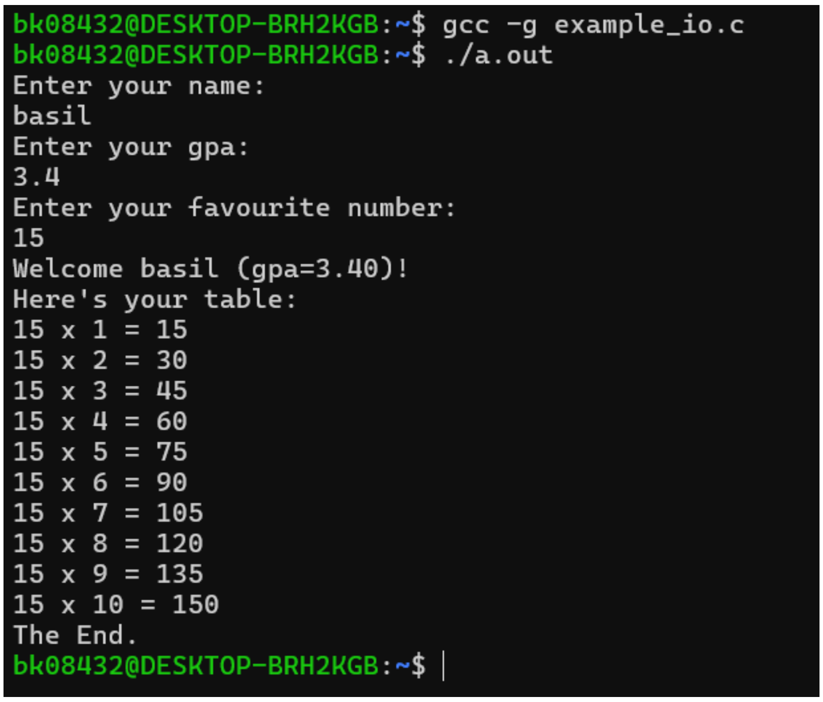
```
bk08432@DESKTOP-BRH2KGB:~$ gcc example_io.c
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$ |
```



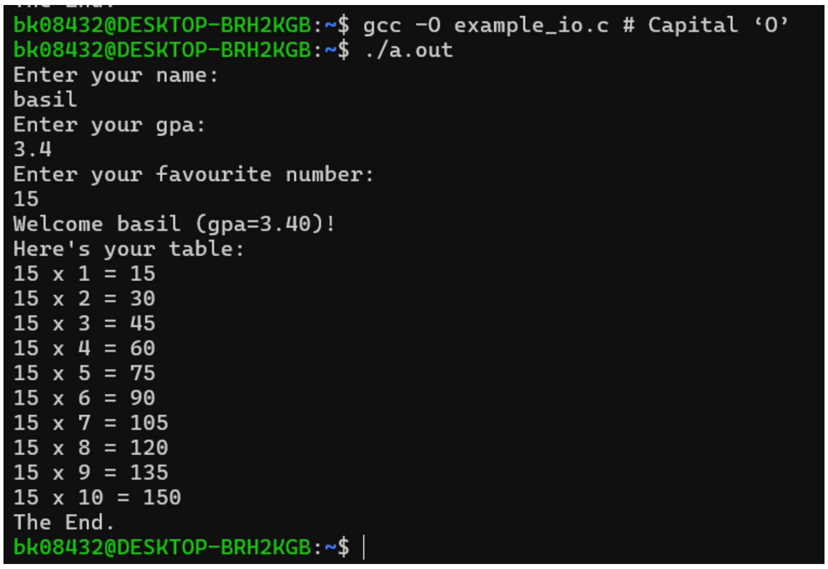
```
bk08432@DESKTOP-BRH2KGB:~$ gcc example_io.c -o example_io # small ‘o
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$
```



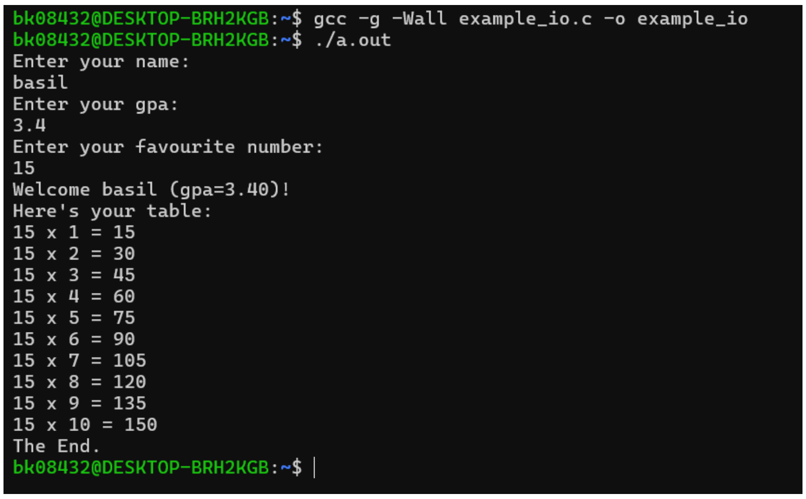
```
bk08432@DESKTOP-BRH2KGB:~$ gcc -Wall example_io.c
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$
```



```
bk08432@DESKTOP-BRH2KGB:~$ gcc -g example_io.c
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$
```



```
bk08432@DESKTOP-BRH2KGB:~$ gcc -O example_io.c # Capital ‘O’
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$
```

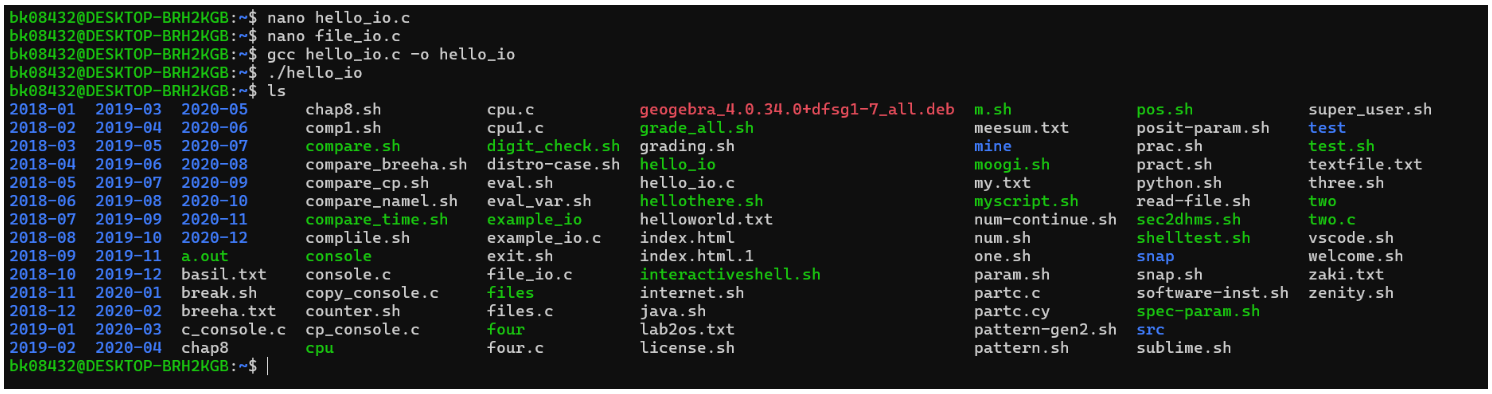


```
bk08432@DESKTOP-BRH2KGB:~$ gcc -g -Wall example_io.c -o example_io
bk08432@DESKTOP-BRH2KGB:~$ ./a.out
Enter your name:
basil
Enter your gpa:
3.4
Enter your favourite number:
15
Welcome basil (gpa=3.40)!
Here's your table:
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150
The End.
bk08432@DESKTOP-BRH2KGB:~$
```

What is the difference between each of the above 6 commands?

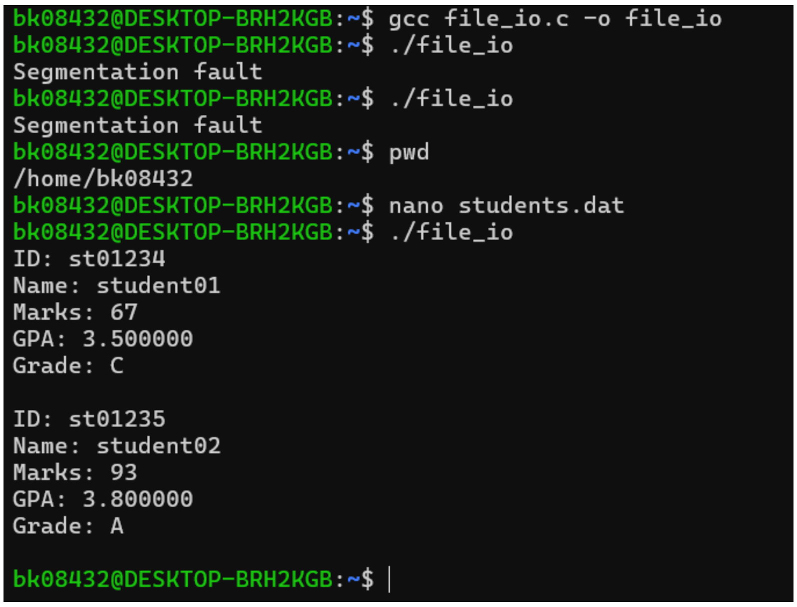
The first command gcc example_io.c compiles the program and generates an executable file with name a.out by default. The second command gcc example_io.c -o example_io specifies the output filename using the small -o flag, which creates an executable called example_io. The third command gcc -Wall example_io.c enables all warning messages, which actually helps to identify potential issues in the code. The fourth command gcc -g example_io.c includes debugging information in the compiled program, useful for debugging with tools like gdb. The fifth command gcc -O example_io.c uses capital -o for improving the efficiency of the generated code. gcc -g -Wall example_io.c -o example_io combines debugging info, all warnings, and custom output filename in one compilation.

Q2)



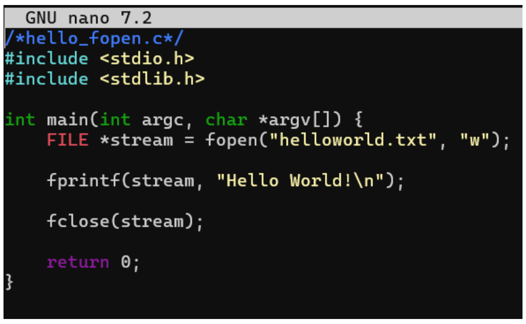
```
bk08432@DESKTOP-BRH2KGB:~$ nano hello_io.c
bk08432@DESKTOP-BRH2KGB:~$ nano file_io.c
bk08432@DESKTOP-BRH2KGB:~$ gcc hello_io.c -o hello_io
bk08432@DESKTOP-BRH2KGB:~$ ./hello_io
bk08432@DESKTOP-BRH2KGB:~$ ls
2018-01  2019-03  2020-05      chap8.sh           cpu.c           geogebra_4.0.34.0+dfsg1-7_all.deb  m.sh             pos.sh            super_user.sh
2018-02  2019-04  2020-06      comp1.sh           cpu1.c          grade_all.sh                       meesum.txt       posit-param.sh    test
2018-03  2019-05  2020-07      compare.sh         digit_check.sh  grading.sh                         mine             prac.sh           test.sh
2018-04  2019-06  2020-08      compare_breeha.sh  distro-case.sh  hello_io                           moogi.sh         pract.sh          textfile.txt
2018-05  2019-07  2020-09      compare_cp.sh      eval.sh         hello_io.c                         my.txt           python.sh         three.sh
2018-06  2019-08  2020-10      compare_namel.sh   eval_var.sh     hellothere.sh                      myscript.sh      read-file.sh      two
2018-07  2019-09  2020-11      compare_time.sh    example_io      helloworld.txt                     num-continue.sh  sec2dhms.sh       two.c
2018-08  2019-10  2020-12      complile.sh        example_io.c    index.html                         num.sh           shelltest.sh      vscode.sh
2018-09  2019-11  a.out        console            exit.sh         index.html.1                       one.sh           snap              welcome.sh
2018-10  2019-12  basil.txt    console.c          file_io.c       interactiveshell.sh                param.sh         snap.sh           zaki.txt
2018-11  2020-01  break.sh     copy_console.c     files           internet.sh                        partc.c          software-inst.sh  zenity.sh
2018-12  2020-02  breeha.txt   counter.sh         files.c         java.sh                            partc.cy         spec-param.sh
2019-01  2020-03  c_console.c  cp_console.c       four            lab2os.txt                         pattern-gen2.sh  src
2019-02  2020-04  chap8        cpu                four.c          license.sh                         pattern.sh       sublime.sh
bk08432@DESKTOP-BRH2KGB:~$ |
```

After running the hello_io.c file there was a helloworld.txt file created with “Hello world” inside it.



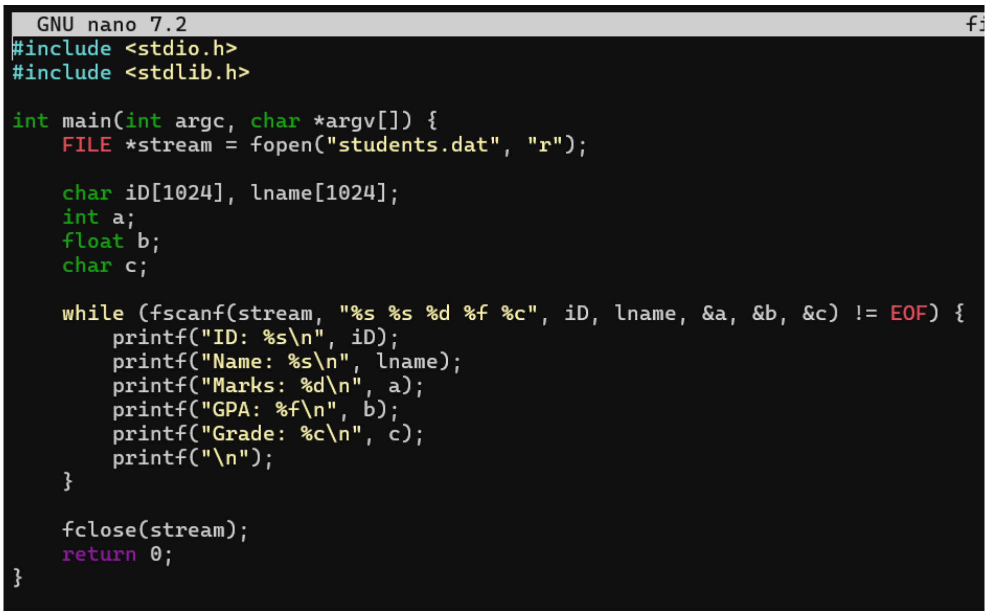
```
bk08432@DESKTOP-BRH2KGB:~$ gcc file_io.c -o file_io
bk08432@DESKTOP-BRH2KGB:~$ ./file_io
Segmentation fault
bk08432@DESKTOP-BRH2KGB:~$ ./file_io
Segmentation fault
bk08432@DESKTOP-BRH2KGB:~$ pwd
/home/bk08432
bk08432@DESKTOP-BRH2KGB:~$ nano students.dat
bk08432@DESKTOP-BRH2KGB:~$ ./file_io
ID: st01234
Name: student01
Marks: 67
GPA: 3.500000
Grade: C

ID: st01235
Name: student02
Marks: 93
GPA: 3.800000
Grade: A

bk08432@DESKTOP-BRH2KGB:~$ |
```

Hello_io.c:



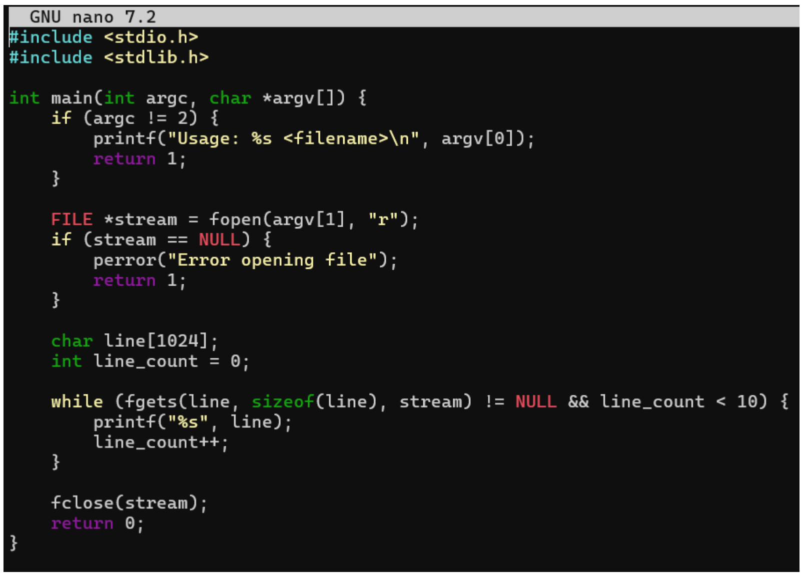
```
 GNU nano 7.2
/*hello_fopen.c*/
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    FILE *stream = fopen("helloworld.txt", "w");

    fprintf(stream, "Hello World!\n");

    fclose(stream);

    return 0;
}
```

File_io.c:



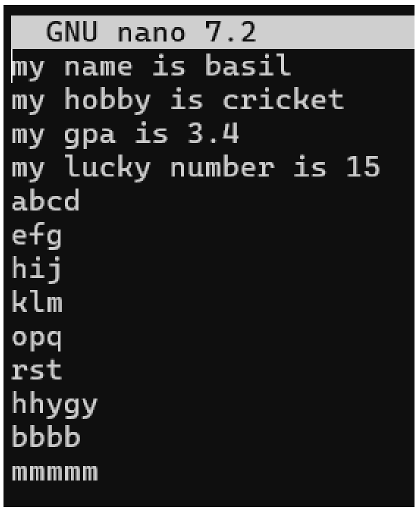
```
 GNU nano 7.2                                                            fi
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    FILE *stream = fopen("students.dat", "r");

    char iD[1024], lname[1024];
    int a;
    float b;
    char c;

    while (fscanf(stream, "%s %s %d %f %c", iD, lname, &a, &b, &c) != EOF) {
        printf("ID: %s\n", iD);
        printf("Name: %s\n", lname);
        printf("Marks: %d\n", a);
        printf("GPA: %f\n", b);
        printf("Grade: %c\n", c);
        printf("\n");
    }

    fclose(stream);
    return 0;
}
```

Changed code:



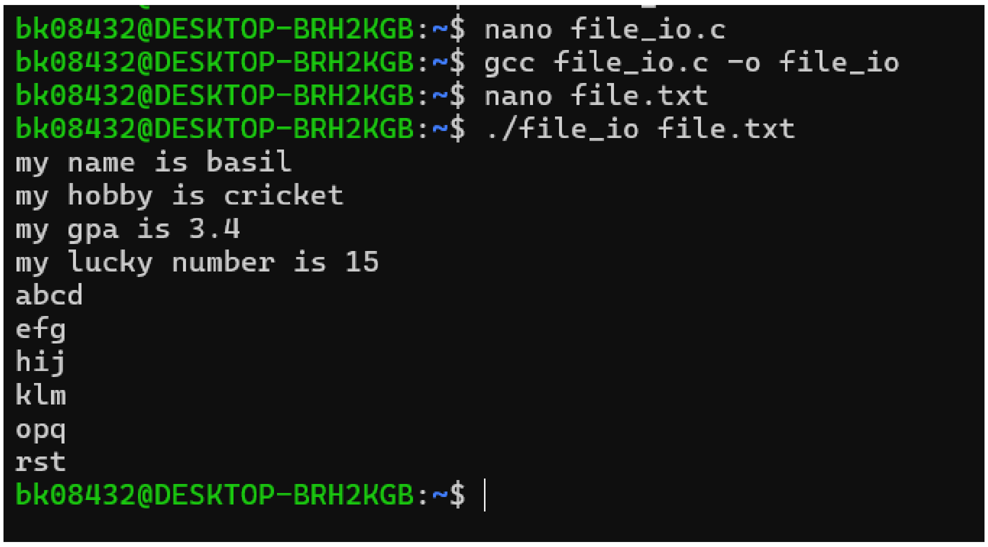
```
 GNU nano 7.2
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *stream = fopen(argv[1], "r");
    if (stream == NULL) {
        perror("Error opening file");
        return 1;
    }

    char line[1024];
    int line_count = 0;

    while (fgets(line, sizeof(line), stream) != NULL && line_count < 10) {
        printf("%s", line);
        line_count++;
    }

    fclose(stream);
    return 0;
}
```

Then I made the file “file.txt” with random text lines shown below:



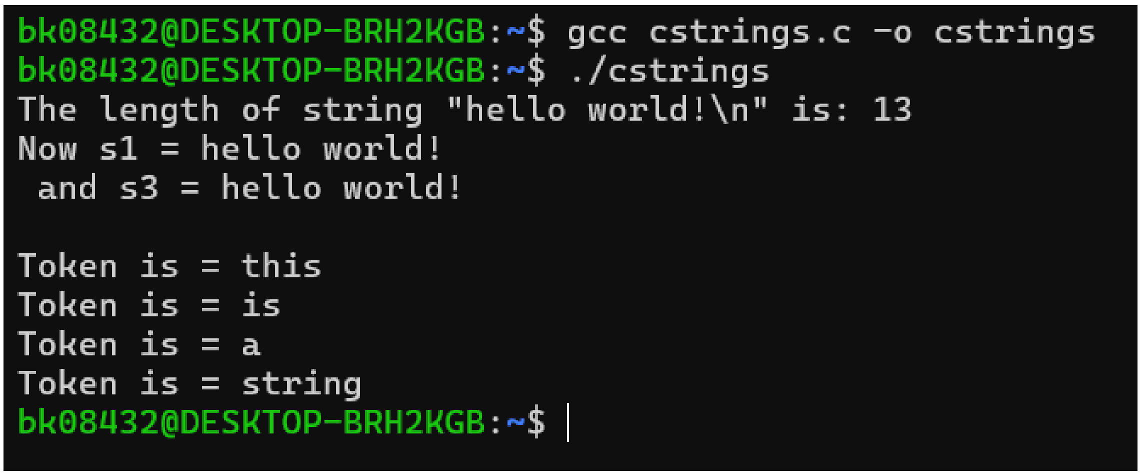
```
  GNU nano 7.2
my name is basil
my hobby is cricket
my gpa is 3.4
my lucky number is 15
abcd
efg
hij
klm
opq
rst
hhygy
bbbb
mmmmm
```

Then I ran the file_io.c and it printed first 10 lines of the file.txt file as shown below:

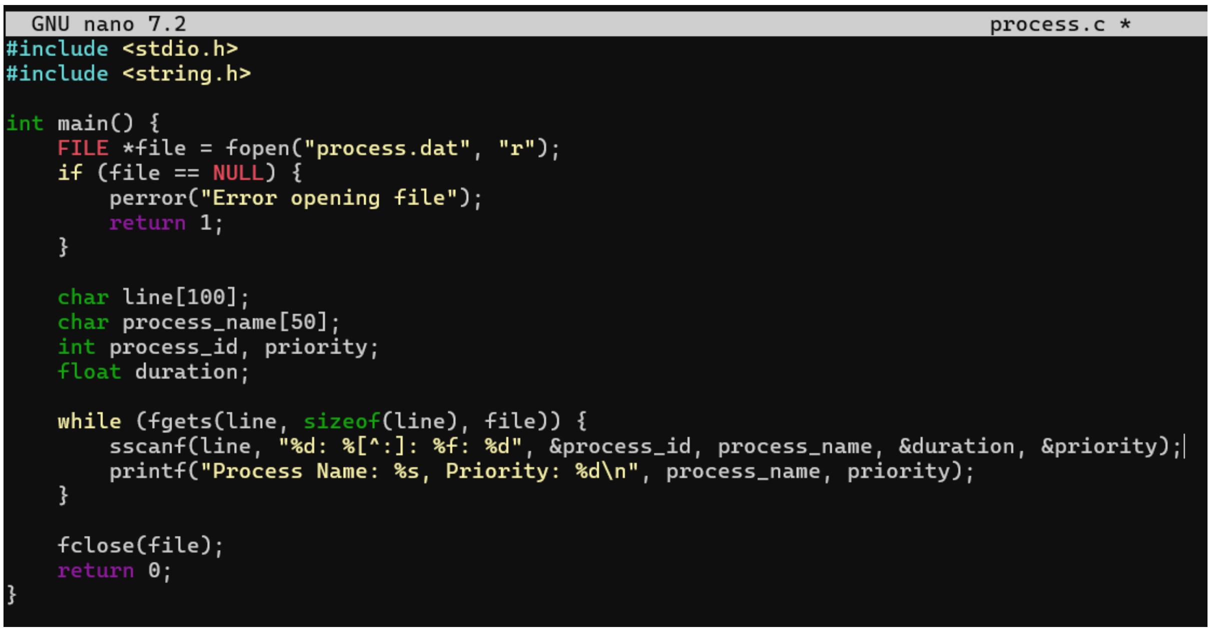


```
bk08432@DESKTOP-BRH2KGB:~$ nano file_io.c
bk08432@DESKTOP-BRH2KGB:~$ gcc file_io.c -o file_io
bk08432@DESKTOP-BRH2KGB:~$ nano file.txt
bk08432@DESKTOP-BRH2KGB:~$ ./file_io file.txt
my name is basil
my hobby is cricket
my gpa is 3.4
my lucky number is 15
abcd
efg
hij
klm
opq
rst
bk08432@DESKTOP-BRH2KGB:~$
```

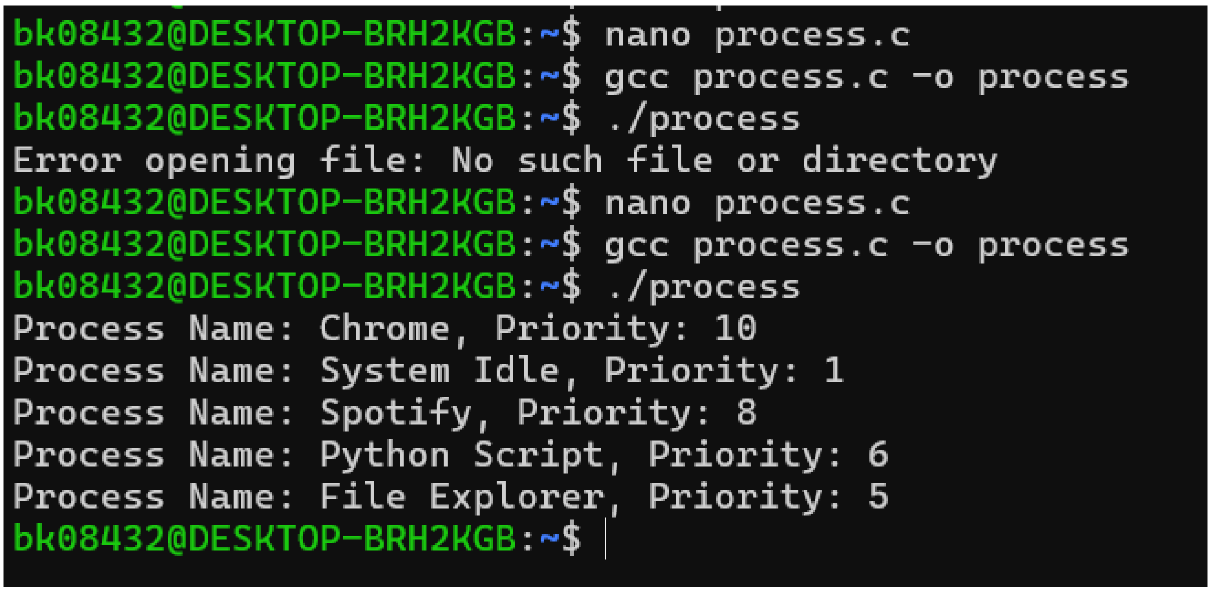
Q3) running the cstring:



```
bk08432@DESKTOP-BRH2KGB:~$ gcc cstrings.c -o cstrings
bk08432@DESKTOP-BRH2KGB:~$ ./cstrings
The length of string "hello world!\n" is: 13
Now s1 = hello world!
 and s3 = hello world!

Token is = this
Token is = is
Token is = a
Token is = string
bk08432@DESKTOP-BRH2KGB:~$
```



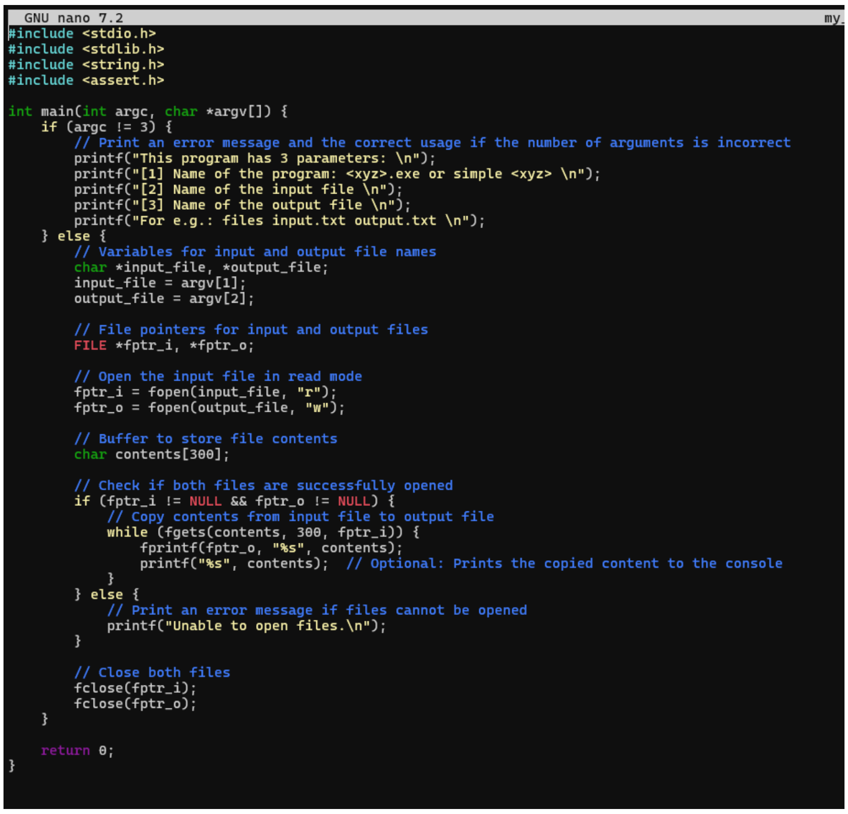
```
  GNU nano 7.2                                                              process.c *
#include <stdio.h>
#include <string.h>

int main() {
    FILE *file = fopen("process.dat", "r");
    if (file == NULL) {
        perror("Error opening file");
        return 1;
    }

    char line[100];
    char process_name[50];
    int process_id, priority;
    float duration;

    while (fgets(line, sizeof(line), file)) {
        sscanf(line, "%d: %[^:]: %f: %d", &process_id, process_name, &duration, &priority);
        printf("Process Name: %s, Priority: %d\n", process_name, priority);
    }

    fclose(file);
    return 0;
}
```



```
bk08432@DESKTOP-BRH2KGB:~$ nano process.c
bk08432@DESKTOP-BRH2KGB:~$ gcc process.c -o process
bk08432@DESKTOP-BRH2KGB:~$ ./process
Error opening file: No such file or directory
bk08432@DESKTOP-BRH2KGB:~$ nano process.c
bk08432@DESKTOP-BRH2KGB:~$ gcc process.c -o process
bk08432@DESKTOP-BRH2KGB:~$ ./process
Process Name: Chrome, Priority: 10
Process Name: System Idle, Priority: 1
Process Name: Spotify, Priority: 8
Process Name: Python Script, Priority: 6
Process Name: File Explorer, Priority: 5
bk08432@DESKTOP-BRH2KGB:~$
```

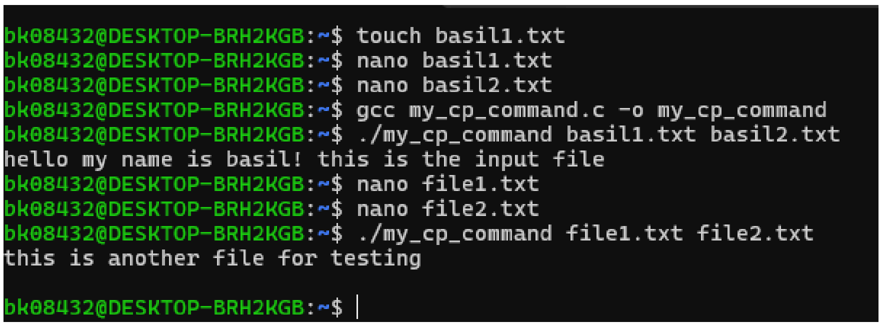
Q4) First I created a basil1.txt and then basil2.txt, the contents of basil 1 were copied into basil2.txt

Code:



```
  GNU nano 7.2                                                                my_
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

int main(int argc, char *argv[]) {
    if (argc != 3) {
        // Print an error message and the correct usage if the number of arguments is incorrect
        printf("This program has 3 parameters: \n");
        printf("[1] Name of the program: <xyz>.exe or simple <xyz> \n");
        printf("[2] Name of the input file \n");
        printf("[3] Name of the output file \n");
        printf("For e.g.: files input.txt output.txt \n");
    } else {
        // Variables for input and output file names
        char *input_file, *output_file;
        input_file = argv[1];
        output_file = argv[2];

        // File pointers for input and output files
        FILE *fptr_i, *fptr_o;

        // Open the input file in read mode
        fptr_i = fopen(input_file, "r");
        fptr_o = fopen(output_file, "w");

        // Buffer to store file contents
        char contents[300];

        // Check if both files are successfully opened
        if (fptr_i != NULL && fptr_o != NULL) {
            // Copy contents from input file to output file
            while (fgets(contents, 300, fptr_i)) {
                fprintf(fptr_o, "%s", contents);
                printf("%s", contents);  // Optional: Prints the copied content to the console
            }
        } else {
            // Print an error message if files cannot be opened
            printf("Unable to open files.\n");
        }

        // Close both files
        fclose(fptr_i);
        fclose(fptr_o);
    }

    return 0;
}
```

Result:



```
bk08432@DESKTOP-BRH2KGB:~$ touch basil1.txt
bk08432@DESKTOP-BRH2KGB:~$ nano basil1.txt
bk08432@DESKTOP-BRH2KGB:~$ nano basil2.txt
bk08432@DESKTOP-BRH2KGB:~$ gcc my_cp_command.c -o my_cp_command
bk08432@DESKTOP-BRH2KGB:~$ ./my_cp_command basil1.txt basil2.txt
hello my name is basil! this is the input file
bk08432@DESKTOP-BRH2KGB:~$ nano file1.txt
bk08432@DESKTOP-BRH2KGB:~$ nano file2.txt
bk08432@DESKTOP-BRH2KGB:~$ ./my_cp_command file1.txt file2.txt
this is another file for testing

bk08432@DESKTOP-BRH2KGB:~$ |
```

Q5) **Explain briefly the C compilation process.**

- **Preprocessing**: firstly, the preprocessor looks at our source code (like hello.c) and handles special instructions, such as #include and #define. For example, if we use #include <stdio.h>, the preprocessor takes the contents of that file and puts it into our code, which can be used later. The output of this step is a modified source code file (for example called hello.i).

  Compilation: Then the compiler takes the preprocessed code and translates it into something closer to machine language. It turns the code into assembly language, which is easier for computers to understand. The output here is an assembly file (like hello.s).

  Assembly: After that, the assembler takes the assembly code and converts it into machine code, which is a binary format that the computer can understand. This step creates an object file (like hello.o), which contains the machine instructions. But this file isn’t a complete executable program yet.

Linking: The linker takes the object file and connects it with other object files or libraries that our program might need, such as functions from the standard C library (like printf). The linker makes sure that all the parts work together and creates the final executable program (like hello). This is the file we can run on our computer.