۱. **الف)(۸ نمره)** اشکالات هر یک از دو برنامه زیر را پیدا کنید.

```c
#include <stdio.h>
int main() {
    FILE *file;
    file = fopen("output.txt", "w");
    if (file == NULL) {
        printf("Could not open file\n");
        return 1;
    }
    if (fputs("Hello, World!\n", file) >= 0) {
        printf("Error writing to file\n");
    }
    return 0;
}
```

پاسخ: چک کردن با EOF غلط است. تابع fputs در صورت موفقیت مقدار غیر منفی و در غیر این صورت EOF را باز میگرداند(یا به طور دقیق تر مقداری منفی را باز میگرداند) (۲ نمره)
fclose در انتها و قبل از return برای اینکه از buffer overflow جلوگیری شود.(۲ نمره)
کد صحیح:

```c
#include <stdio.h>
int main() {
    FILE *file;
    file = fopen("output.txt", "w");
    if (file == NULL) {
        printf("Could not open file\n");
        return 1;
    }
    if (fputs("Hello, World!\n", file) == EOF) {
        printf("Error writing to file\n");
    }
    fclose(file);
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    FILE *file;
    file = fopen("data.txt", "a"); // Appending mode
    if (file == NULL) {
        printf("Could not open file\n");
        return 1;
    }
    fprintf(file, "New line\n");
    fclose(file);
    char buffer[100];
    file = fopen("data.txt", "r");
    fgets(buffer, sizeof(buffer), file);
    printf("First line: %s\n", buffer);
    fclose(file);
    return 0;
}
```

پاسخ: پس از اینکه فایل برای append کردن باز شد و مجدد بسته شده و سپس برای read کردن مجدد باز شد باید دوباره چک کنیم که فایل درست باز شده یا نه.(۲نمره)

کد شرایطی را در نظر نگرفته که fgets با شکست مواجه شود.(۲ نمره)

کد صحیح:

```c
#include <stdio.h>
int main() {
    FILE *file;
    file = fopen("data.txt", "a"); // Appending mode
    if (file == NULL) {
        printf("Could not open file\n");
        return 1;
    }
    fprintf(file, "New line\n");
    fclose(file);
    char buffer[100];
    file = fopen("data.txt", "r");
    if (file == NULL) {
        printf("Could not open file\n");
        return 1;
    }
    fgets(buffer, sizeof(buffer), file);
    if (fgets(buffer, sizeof(buffer), file) != NULL) {
        printf("First line: %s\n", buffer);
    }
    else {
        printf("Error reading from file\n");
    }
    fclose(file);
    return 0;
```

```
}
```

ب)(۲ نمره) خروجی برنامه زیر را مشخص کنید.

```
int main() {
    FILE* my_file = fopen("file.txt", "w");
    fputs("Height", my_file);
    fseek(my_file, 2, SEEK_SET);
    fputs("yes", my_file);
    fseek(my_file,4, SEEK_SET);
    fputs("very", my_file);
    fseek(my_file, 8, SEEK_SET);
    fputs("one", my_file);
    fseek(my_file, 11, SEEK_SET);
    fclose(my_file);
}
```

پاسخ:

Height>Heye>Heyevery>Heyeveryone

۲. **(۱۰ نمره)** می خواهیم تابعی به زبان c بنویسیم که فایلی از نوع Text را که نام آن را به عنوان ورودی دریافت می کند، باز کرده و همه کدهای اسکی مربوط به "next line" را با "," جایگزین کرده سپس نتیجه را در فایل جدیدی که نام آن را نیز به عنوان ورودی دریافت کرده، بنویسد. مقدار برگشتی این تابع مقدار ۰ در صورت موفقیت و مقدار ۱ در صورت عدم موفقیت می باشد.
پروتوتایپ این تابع به صورت زیر است.

```
int replaceNextlinesWithCommas(const char *in, const char *out){
…}
```

فایل ورودی را input.txt و فایل خروجی را output.txt در نظر بگیرید.

پاسخ:

```
#include <stdio.h>

int replaceNextlinesWithCommas(const char *in, const char *out) {
    FILE *inputFile = fopen(in, "r");//input file only for reading
    FILE *outputFile = fopen(out, "w");//output file for writing
    if (inputFile == NULL || outputFile == NULL) {//checking if files were
opened successfully
        printf("Error opening file");
        return 1;
    }
    char c;
    while ((c = fgetc(inputFile)) != EOF) {//checking for not reaching the end
of the file
        if (c == '\n') {
```

```
            fputc(',', outputFile);//replacing \n with ,
        } else {
            fputc(c, outputFile);
        }
    }
    fclose(inputFile);
    fclose(outputFile);
    return 0;
}

int main() {
    const char *inputFilename = "input.txt";
    const char *outputFilename = "output.txt";
    int func=replaceNextlinesWithCommas(inputFilename, outputFilename);
    if (func==0){
        printf("Replaced newlines with commas successfully.\n");
    }
    return 0;
}
```

۳. **(۱۲ نمره)** فایلی با نام name.txt داریم که حاوی اسامی دانشجویان یک کلاس است. نام هر دانشجو در خطی جدا نوشته شده است. برنامه ای بنویسید نام هر دانشجو را به کاربر نشان داده و نمره دانشجو را از کاربر بگیرد و نمره به علاوه نام را مطابق مثال زیر در فایل جدیدی با نام names_plus_grades.txt بنویسد. .(فرض کنید کاربر عددی بین ۰ تا ۱۰۰ را وارد میکند)

به مثال زیر توجه کنید:

names.txt
Fariba:
Mohsen:
Kasra:
Amin:

Names_plus_grades.txt:
Fariba:97
Mohsen:78
Kasra:66
Amin:55

(راهنمایی: در صورتی که از دستور fgets استفاده میکنید میتوانید از تابع زیر استفاده نمایید:

```
int removeNewLine(const char *str){
…}
```

(

پاسخ:

```c
#include <stdio.h>
//Function to remove newline characters to avoid having a \n at the end of
each name
void removeNewline(char *str) {
    while (*str) {
        if (*str == '\n') {
            *str = '\0';
        }
        str++;
    }
}
int main() {
    FILE *inputFile, *outputFile;
    char name[100]; {//maximum letter a name can have
    int grade;
    inputFile = fopen("names.txt", "r");
    if (inputFile == NULL) {//checking if the file was successfully opened
        printf("Could not open input file\n");
        return 1;
    }
    outputFile = fopen("names_plus_grades.txt", "w");
    //checking if the file was successfully opened
    if (outputFile == NULL) {
        printf("Could not open output file\n");
        fclose(inputFile);
        return 1;
    }
    //Reading each student's name
    while (fgets(name, sizeof(name), inputFile) != NULL) {
        //Removing the newline character
        removeNewline(name);
        printf("Enter grade for %s: ", name);
        scanf("%d", &grade);
        //Writing the student's name and grade to the output file
        fprintf(outputFile, "%s:%d\n", name, grade);
    }
    fclose(inputFile);
    fclose(outputFile);
    return 0;
}
```

توضیحات: تابع removeNewLine به این جهت گذاشته شده که دستور fgets در انتها وقتی رشته را دریافت کرد \n میگذارد که در ادامه اگر \n را حذف نکنیم مشکل ساز میشود. دقت شود که تابع gets این مشکل را ندارد اما میتواند موجب buffer overflow شود.

۴. (۱۲ نمره) برنامه ای بنویسید که دو فایل را خط به خط با هم ادغام کند. در صورتی که تعداد خطوط یک فایل با دیگری یکسان نبود، پس از ادغام خط به خط دو فایل، خطوط مانده از فایل بلندتر را به انتهای فایل اضافه کند.(برای سادگی فرض کنید خط آخر فایل کوتاهتر خالی است)

مثال:

input1.txt:
spring
summer
autumn
winter

---

input2.txt:
March
June
September
December
January
February

---

merged.txt:
spring
March
summer
June
autumn
September
winter
December
January
February

پاسخ:

```c
#include <stdio.h>

int main() {
    FILE *file1, *file2, *mergedFile;
    char line1[256], line2[256];
    int endOfFile1 = 0, endOfFile2 = 0;
    file1 = fopen("input1.txt", "r");
    //checking weather the files were successfully opend or not
    if (file1 == NULL) {
        printf("Error opening input1.txt");
        return 1;
```

```
    }
    file2 = fopen("input2.txt", "r");
    if (file2 == NULL) {
        printf("Error opening input2.txt");
        fclose(file1);
        return 1;
    }
    mergedFile = fopen("merged.txt", "w");
    if (mergedFile == NULL) {
        printf("Error opening merged.txt");
        fclose(file1);
        fclose(file2);
        return 1;
    }
    //reading lines from both files one by one then writing them in the merged
file
    while (1) {
        if (fgets(line1, sizeof(line1), file1) != NULL) {
            fputs(line1, mergedFile);
        } else {
            endOfFile1 = 1;//reaching the end of input1.txt
        }
        if (fgets(line2, sizeof(line2), file2) != NULL) {
            fputs(line2, mergedFile);
        } else {
            endOfFile2 = 1;//reaching the end of input2.txt
        }
        //break if both files have reached the end
        if (endOfFile1 && endOfFile2) {
            break; }
    }
    fclose(file1);
    fclose(file2);
    fclose(mergedFile);
    return 0;
}
```

۵. **امتیازی** (۱۵ نمره)** برنامه ای بنویسید که از کاربر رشته ای حرفی (string) را دریافت میکند و در صورت وجود آن رشته حرفی در فایل، شماره خطوطی که آن رشته را دارا بوده در فایل دیگری مینویسد.
(استفاده از <string.h> مجاز میباشد.)

نکته: دقت کنید اگر به طور مثال کاربر کلمه ی "hi" را وارد کند نباید کلمه ی "him" به عنوان یکی از کلمات در خروجی شمرده شود.

پاسخ:

```
#include <stdio.h>
#include <string.h>

//Function to remove the newline character from the end of a string
void removeNewline(char *str) {
```

```
    int len = strlen(str);
    if (len > 0 && str[len - 1] == '\n') {
        str[len - 1] = '\0'; //Replacing the newline character with a null
    }
}

//Helper function to check if a character is a word boundary
int isWordBoundary(char c) {
    //boundries: spaces, tabs, newlines, carriage returns, null,punctuation
    return (c == ' ' || c == '\t' || c == '\n' || c == '\r' || c == '\0' || c
== '.' || c == ',' || c == '!' || c == '?' || c == ';' || c == ':');
}

//Function to count occurrences of the name as a whole word in the line
int countWordInLine(const char *line, const char *name) {
    int nameLen = strlen(name); // Length of the name to search for
    int lineLen = strlen(line); // Length of the current line
    int count = 0; // Count of occurrences
    for (int i = 0; i <= lineLen - nameLen; i++) {
        //Checking if the name is a whole word at the current position
        if ((i == 0 || isWordBoundary(line[i - 1])) &&
            (strncmp(&line[i], name, nameLen) == 0) &&
            (line[i + nameLen] == '\0' || isWordBoundary(line[i + nameLen])))
{
            count++; // Increment the count if a whole word match is found
            i += nameLen - 1; // Move the index forward to avoid overlapping
matches
        }
    }
    return count; //Return the total count of occurrences in the line
}

//Function to check the name in the file and count occurrences
void checkNameInFile(const char *filename) {
    char name[100];
    printf("Enter the name you want to search for: ");
    fgets(name, sizeof(name), stdin); //Read the name from user input
    removeNewline(name); // Remove the newline character from the input

    FILE *file = fopen(filename, "r"); //Open the file for reading
    if (file == NULL) {
        printf("Error opening file"); //Print an error message if the file
wasn't be opened
        return;
    }

    char line[256]; //maximum lines is considered 256
    int totalCount = 0; //number of occurrences
```

```
    //Reading each line from the file
    while (fgets(line, sizeof(line), file) != NULL) {
        totalCount += countWordInLine(line, name); //Adding the number of
occurrences in the current line to the total count
    }

    // Print the result
    if (totalCount > 0) {
        printf("The name '%s' was found %d times in the file.\n", name,
totalCount);
    } else {
        printf("The name '%s' was not found in the file.\n", name);
    }

    fclose(file); //closing the file
    }
    int main() {
        const char *filename = "input.txt";
        checkNameInFile(filename);
        return 0;
    }
```

توضیحات: تابع removeNewLine به این جهت گذاشته شده که دستور fgets در انتها وقتی رشته را دریافت کرد \n میگذارد که در ادامه اگر \n را حذف نکنیم مشکل ساز میشود. دقت شود که تابع gets این مشکل را ندارد اما میتواند موجب buffer overflow شود.

ضرورت تابع isWordBoundary به علت این است که اگر به طور مثال کلمه ی "hi" را کاربر وارد میکند کلمه ای مانند "him" به عنوان یکی از کلمات در نظر گرفته نشود.

در ادامه الگوریتم به این صورت است که در هر خط تعداد رشته ای که کاربر وارد کرده را شمرده و آن ها را با هم جمع میکنیم.