R6: Files

General Instructions:

1- Create a C project in eclipse called R6

2- Extract the file R6.zip into a folder of your choice at your machine. Copy the files into the R6 project.

3- Rename the file: “R6_template.txt” to “R6.c”. Enter your credentials on top of the file.

4- Make sure that all the input files (.txt files) are on the same directory level of your code.

5- Follow the videos posted by the instructor on how to complete the tasks presented in this worksheet.

6- To test your code, select the proper function in the file: “R6_test.c”. Compare your results with those presented in “R6_output.txt”. Your results should exactly match the output.

7- Submit R6.c file.

General comments:

In this course, whenever you receive a filename (string) as an input parameter then it means you are expected to do all file handling (e.g. opening and closing) inside the function. However, if you receive a file handler (FILE*) then assume that the opening and closing happen outside the function.

Also, whenever you open a file, you need to perform the following error checking:

```
if (fptr == NULL) { //fptr is the file pointer
      printf("Error(<function_name>): Could not open file %s\n", filename);
      return -1; //or return or return NULL
}
```

Task 1 Write to a File

Implement the following function:

```
void write_nums(const char *filename, int start, int end);
```

The functions receives a character pointer (string) which represent a filename. The function also receives two other integers: start and end. The function writes the numbers from start to end, inclusive both ends, each number in a separate line.

Task 2 Read from a file:

Implement the following function:

```
int read_nums(const char *filename, int *array);
```

The functions receives a character pointer (string) which represent a filename. The function also receives an integer pointer, representing an array of integers of size MAX. The function reads the given file and store the numbers in the array. Assume that the file is formatted such that each line contain a single number (similar to Task 1). The function returns a counter representing how many numbers were read from the file.

Task 3 Extract Minutes:

Implement the following function:

```
int extract_minutes(char *filename, unsigned int *minutes_array,
const int size);
```

The functions receives a character pointer (string) which represent a filename. and a pointer to an integer array along with its size.

The function reads the given file which has times in the following format: HH:MM:SS. Each time record appear in a separate line.

The function extract the minutes from each line (MM) and store it in the given minutes_array. The function keeps reading from the file until it reaches end of file, or the minutes_array is full.

The function returns the number of time records read from the file.

Task 4 Print Record (Version 1):

Implement the following function:

```
void print_record_info1(char *record);
```

The functions receives a character pointer (string) which represent student record. Each record is structured as the following:

“[registration_year-birth_month-birth_day-birth_year-first_name last_name]”

An example is: `"[2018-June-14-1994-David Smith]"`

The function uses strtok function to the extract the student information and then print the results in the following format:

```
Registration year = 2018
Birthday = June 14,1994
Name = David Smith
```

Task 5 Print Record (Version 2):

Implement the following function:

```
void print_record_info2(char *record);
```

The function is similar to Task 4, except that the record is formatted with spaces instead of a dash character

“[registration_year birth_month birth_day birth_year first_name last_name]”

An example is: `"[2018 June 14 1994 David Smith]"`

The function uses sscanf function to the extract the student information and then print the results in the following format:

```
Registration year = 2018
Birthday = June 14,1994
Name = David Smith
```

Task 6 Print Record (Version 3):

Implement the following function:

```
void print_record_info3(char *record);
```

The function is similar to Task 4 and 5, except that the record is formatted as the following example:

An example is: `"record:2018-June,14,1994-(David Smith)"`

Use a combination of strtok and sscanf to extract the student information, and print it as follows:

```
Registration year = 2018
Birthday = June 14,1994
Name = David Smith
```

Task 7 Update a file

Implement the following function:

```
void modify_grades(char *filename, int bonus);
```

The function receives a filename and an integer representing a grade bonus.

The file contain a list of student grades adjacent to each other, like the following:

```
[111111111,28][222222222,30][333333333,33][444333333,36]
```

Each student record is [XXXXXXXXX,YY], where the X part is a 9-digit student ID, and the Y part is a two-digit student grade.

The function updates each student record by the given bonus. In the above example, if the bonus was 2, then the output will be:

```
[111111111,30][222222222,32][333333333,35][444333333,38]
```

In your implementation, use fseek along with its two reference pointers: SEEK_CUR and SEEK_SET.