C Programming II
2024 Spring
Homework 02

Instructor: Po-Wen Chi

Due: 2024.xx.xx PM 11:59

Policies:

- **Zero tolerance** for late submission.
- **Plagiarism is not allowed.** Both source and copycat will be **zero**.
- You need to prepare a README file about how to make and run your program. Moreover, you need to provide your name and your student ID in the README file.
  - Your Name and Your ID.
  - The functional description for each code.
  - Anything special.
- Please pack all your submissions in one zip file.
- For convenience, your executable programs must be named following the rule hwXXYY, where the red part is the homework number and the blue part is the problem number. For example, hw0102 is the executable program for homework #1 problem 2.
- I only accept **PDF**. MS Word is not allowed.
- Do not forget your Makefile. For convenience, each assignment needs only one Makefile.

1 Subtitle Player (20 pts)

When you watch an English video, do you need its subtitle? This time, I want you to develop a subtitle player. The subtitle format we use in this problem is **Advanced SubStation Alpha（ASS）** . Please see the following link for its detail.

https://zh.wikipedia.org/zh-tw/SubStation_Alpha#Advanced_SubStation_Alpha%EF%BC%88ASS%E5%AD%97%E5%B9%95%EF%BC%89

I will also provide you a subtitle example on the website. The program usage is as follows.

```
$ ./hw0201
Please enter the file name: test.ass
Time Shift ( -10 ~ 10 ): 0                    // <-- integer
Speed (0.25,0.5,0.75,1,1.25,1.5,1.75,2): 1    // <-- only support these
    candidate
```

The subtitle is the example on the website. After the above input and 25.06 seconds, your should output the following string.

```
... Three minutes.
```

After 3.8 seconds, please **clear the screen**. Then, after 0.008 second, you should print the following string.

```
Oui, Chef. Turn on the stove. Come on. Hurry up. Oui, Chef.
```

Your program should repeat this operation till the end of file. When printing strings, do not forget to print them with **colors**[1].

2 Premier League (20 pts)

Given a Premier League match data for some season, please print the final table of that season. You can download data from the following site.

https://github.com/datasets/football-datasets/tree/master/datasets/premier-league

```
./hw0202
Please enter the data file name: season-1819.csv
    Team                   W    D    L    GF    GA    GD    Pts
01) Manchester City FC     32   2    4    95    23    +72    98
02) Liverpool FC           30   7    1    89    22    +67    97
03) Chelsea FC             21   9    8    63    39    +24    72
...
19) Fulham FC              7    5    26   34    81    -47    26
20) Huddersfield Town FC   3    7    28   22    76    -54    16
```

Note that your output must be aligned.

3 Wordle Solver (20 pts)

Do you know what **Wordle** is? If not, please access the following link and play the Game.

https://www.nytimes.com/games/wordle/index.html

As you know, my English is poor. Therefore I want you to develop a tool for me to win this game. I will show you what I want.

1. First, you need an English dictionary. You can install **hunspell-en-us** in Ubuntu and get the English dictionary **en_US.dic**. Or you can directly download it from the following link.
   https://cgit.freedesktop.org/libreoffice/dictionaries/tree/en

2. Find all possible 5-letter words from the dictionary.

[1] According to the PrimaryColour defined in Styles.

3. Propose the most possible candidate word to the user.

    - You must use the letter frequency table from the following link to calculate the summation of 5-letter frequency. You should use the text one. The 5-letter word in the candidate with the highest frequency is the most possible one.
      https://en.wikipedia.org/wiki/Letter_frequency

4. Let the user to input the feedback from the Wordle site. The feedback will be "XXXXX" where X can be:

    - G: in the word and in the correct spot.
    - Y: in the word but in the wrong spot.
    - B: not in the word in any spot.

For your convenience,

- You do not need to provide the dictionary file when submitting your homework. Our TAs will put a dictionary file in your program directory and you can simply open the dictionary file directly in your program.
- Undoubtedly, there is no complete dictionary in this world. In this problem, you can just treat the dictionary as the complete one.
- You do not need to care about any variations of the word.
- The input should be case insensitive.

The program interface is as follows. Note that this is from my playing record and I do not do frequency analysis when playing.

```
./hw0203
Advice:   AUDIO   // <-- Advice from your program
Feedback: YBYBB   // <-- User input
Advice:   SHADE   // <-- Advice from your program
Feedback: GGGGG   // <-- User input
Congratulations!!
```

4 BMP (20 pts)

I want you to develop a program to distort a BMP file with a given angle.

```
$ ./hw0204
Please input a BMP file: doraemon.bmp
Please input the output BMP file name: doraemon_out.bmp
Angle (0-90): 45
```

The distortion is as in figure 1.

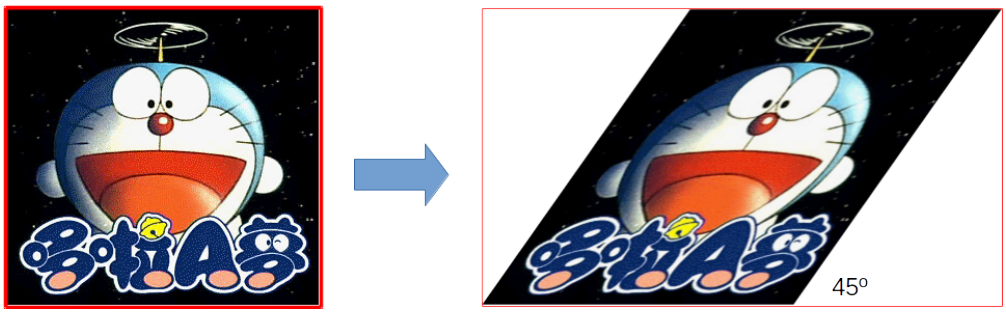

Figure 1: Distortion Example. Note that the boundary red line should not be presented. The additional part should be colored with white.

5 Large Language Model (20 pts)

In 2022, ChatGPT garnered widespread attention; in 2023, following the release of the foundational model **LLaMa**, thousands of Large Language Models (LLMs) emerged, signaling the rise of generative AI. This trend has continued into 2024.

While the global community races to develop increasingly powerful models reliant on massive GPU clusters with thousands of GPU, a segment of researchers and the open-source community is taking a different approach. Their goal is to make large language models accessible on a wide range of devices, including smartphones, with or without specialized acceleration hardware.

A standout project in this effort is LLaMa.cpp. This project utilizes CPU acceleration technologies (e.g., NEON, AVX) and GPU acceleration frameworks (e.g., Metal, CUDA) to facilitate model inference on devices like laptops and smartphones. It employs the GGML compute library at its core, which is a pure C library designed for executing machine learning computations efficiently on CPU, GPU, and other devices.

LLaMa.cpp uses a special format to store the metadata and the model weights, called GGUF. Fig. 2 is a beautiful visualization of GGUF v3, provided in the official documentation.

Don't worry, I am not asking you to build an inference engine in pure C. (Some people really do so!) Instead, you are going to implement a GGUF file reader which is able to read the **model.gguf** in the current directory and output the metadata, the tensors, and the total parameters (summed up by all tensors) of the model.

The structure of GGUF is as simple as what you see in the above image:

```c
struct gguf_file_t {
    // The header of the file.
    gguf_header_t header;

    // Tensor infos, which can be used to locate the tensor data.
    gguf_tensor_info_t tensor_infos[header.tensor_count];

    // Other data that you don't need to worry about this time.
    uint8_t other_binary_things[];
};
```

Figure 2: Model visualization: model.gguf

Then, what are **gguf_header_t** and **gguf_tensor_info_t**? Here are the structs that are used internally in GGML, where you can see the real usage of union and enum that you are familiar with!

```c
struct gguf_header_t {
    // Magic number to announce that this is a GGUF file.
    // Must be `GGUF` at the byte level: `0x47` `0x47` `0x55` `0x46`.
    // Your executor might do little-endian byte order, so it might be
    // check for 0x46554747 and letting the endianness cancel out.
    // Consider being *very* explicit about the byte order here.
    uint32_t magic;
    // The version of the format implemented.
    // Must be `3` for version described in this spec, which introduces big-endian support.
    //
    // This version should only be increased for structural changes to the format.
    // Changes that do not affect the structure of the file should instead update the metadata
    // to signify the change.
    uint32_t version;
    // The number of tensors in the file.
    // This is explicit, instead of being included in the metadata, to ensure it is always present
```

```
    // for loading the tensors.
    uint64_t tensor_count;
    // The number of metadata key-value pairs.
    uint64_t metadata_kv_count;
    // The metadata key-value pairs.
    gguf_metadata_kv_t metadata_kv[metadata_kv_count];
};

struct gguf_tensor_info_t {
    // The name of the tensor. It is a standard GGUF string, with the caveat
   that
    // it must be at most 64 bytes long.
    gguf_string_t name;
    // The number of dimensions in the tensor.
    // Currently at most 4, but this may change in the future.
    uint32_t n_dimensions;
    // The dimensions of the tensor.
    uint64_t dimensions[n_dimensions];
    // The type of the tensor.
    ggml_type type;
    // The offset of the tensor's data in this file in bytes.
    //
    // This offset is relative to `tensor_data`, not to the start
    // of the file, to make it easier for writers to write the file.
    // Readers should consider exposing this offset relative to the
    // file to make it easier to read the data.
    //
    // Must be a multiple of `ALIGNMENT`. That is, `align_offset(offset) ==
   offset`.
    uint64_t offset;
};

struct gguf_metadata_kv_t {
    // The key of the metadata. It is a standard GGUF string, with the
   following caveats:
    // - It must be a valid ASCII string.
    // - It must be a hierarchical key, where each segment is `
   lower_snake_case` and separated by a `.`.
    // - It must be at most 2^16-1/65535 bytes long.
    // Any keys that do not follow these rules are invalid.
    gguf_string_t key;

    // The type of the value.
    // Must be one of the `gguf_metadata_value_type` values.
    gguf_metadata_value_type value_type;
    // The value.
    gguf_metadata_value_t value;
};

// A string in GGUF.
struct gguf_string_t {
    // The length of the string, in bytes.
    uint64_t len;
    // The string as a UTF-8 non-null-terminated string.
```

```
    char string[len];
}

union gguf_metadata_value_t {
    uint8_t uint8;
    int8_t int8;
    uint16_t uint16;
    int16_t int16;
    uint32_t uint32;
    int32_t int32;
    float float32;
    uint64_t uint64;
    int64_t int64;
    double float64;
    bool bool_;
    gguf_string_t string;
    struct {
        // Any value type is valid, including arrays.
        gguf_metadata_value_type type;
        // Number of elements, not bytes
        uint64_t len;
        // The array of values.
        gguf_metadata_value_t array[len];
    } array;
};

enum ggml_type: uint32_t {
    GGML_TYPE_F32  = 0,
    GGML_TYPE_F16  = 1,
    GGML_TYPE_Q4_0 = 2,
    GGML_TYPE_Q4_1 = 3,
    // GGML_TYPE_Q4_2 = 4, support has been removed
    // GGML_TYPE_Q4_3 (5) support has been removed
    GGML_TYPE_Q5_0 = 6,
    GGML_TYPE_Q5_1 = 7,
    GGML_TYPE_Q8_0 = 8,
    GGML_TYPE_Q8_1 = 9,
    // k-quantizations
    GGML_TYPE_Q2_K = 10,
    GGML_TYPE_Q3_K = 11,
    GGML_TYPE_Q4_K = 12,
    GGML_TYPE_Q5_K = 13,
    GGML_TYPE_Q6_K = 14,
    GGML_TYPE_Q8_K = 15,
    GGML_TYPE_I8,
    GGML_TYPE_I16,
    GGML_TYPE_I32,
    GGML_TYPE_COUNT,
};

enum gguf_metadata_value_type: uint32_t {
    // The value is a 8-bit unsigned integer.
    GGUF_METADATA_VALUE_TYPE_UINT8 = 0,
    // The value is a 8-bit signed integer.
```

```
    GGUF_METADATA_VALUE_TYPE_INT8 = 1,
    // The value is a 16-bit unsigned little-endian integer.
    GGUF_METADATA_VALUE_TYPE_UINT16 = 2,
    // The value is a 16-bit signed little-endian integer.
    GGUF_METADATA_VALUE_TYPE_INT16 = 3,
    // The value is a 32-bit unsigned little-endian integer.
    GGUF_METADATA_VALUE_TYPE_UINT32 = 4,
    // The value is a 32-bit signed little-endian integer.
    GGUF_METADATA_VALUE_TYPE_INT32 = 5,
    // The value is a 32-bit IEEE754 floating point number.
    GGUF_METADATA_VALUE_TYPE_FLOAT32 = 6,
    // The value is a boolean.
    // 1-byte value where 0 is false and 1 is true.
    // Anything else is invalid, and should be treated as either the model
   being invalid or the reader being buggy.
    GGUF_METADATA_VALUE_TYPE_BOOL = 7,
    // The value is a UTF-8 non-null-terminated string, with length prepended.
    GGUF_METADATA_VALUE_TYPE_STRING = 8,
    // The value is an array of other values, with the length and type
   prepended.
    ///
    // Arrays can be nested, and the length of the array is the number of
   elements in the array, not the number of bytes.
    GGUF_METADATA_VALUE_TYPE_ARRAY = 9,
    // The value is a 64-bit unsigned little-endian integer.
    GGUF_METADATA_VALUE_TYPE_UINT64 = 10,
    // The value is a 64-bit signed little-endian integer.
    GGUF_METADATA_VALUE_TYPE_INT64 = 11,
    // The value is a 64-bit IEEE754 floating point number.
    GGUF_METADATA_VALUE_TYPE_FLOAT64 = 12,
}
```

The output format matters. For some long fields, please ensure they fit into an **80-column width**.

```
$ ./hw0205
GGUF: true
Parameters: 1,100,048,384

Metadata                                Value
version                                 2
tensor_count                            201
kv_count                                21
general.architecture                    llama
general.name                            models
general.file_type                       2
general.quantization_version            2
llama.context_length                    2048
llama.embedding_length                  2048
llama.block_count                       22
llama.feed_forward_length               5632
llama.rope.dimension_count              64
llama.rope.freq_base                    10000
llama.attention.head_count              32
llama.attention.head_count_kv           4
```

```
llama.attention.layer_norm_rms_epsilon  0.00000999999747378752
tokenizer.ggml.model                    llama
tokenizer.ggml.tokens                   [<unk>, <s>, </s>, <0x00>, <0x01>,
    ...]
tokenizer.ggml.scores                   [0, 0, 0, 0, 0, ...]
tokenizer.ggml.token_type               [2, 3, 3, 6, 6, ...]
tokenizer.ggml.bos_token_id             1
tokenizer.ggml.eos_token_id             2
tokenizer.ggml.padding_token_id         2

Tensors                                 Shape           Precision
blk.0
    .attn_k.weight                      [2048,256]      Q4_0
    .attn_norm.weight                   [2048]          F32
    .attn_output.weight                 [2048,2048]     Q4_0
    .attn_q.weight                      [2048,2048]     Q4_0
    .attn_v.weight                      [2048,256]      Q4_0
    .ffn_down.weight                    [5632,2048]     Q4_0
    .ffn_gate.weight                    [2048,5632]     Q4_0
    .ffn_norm.weight                    [2048]          F32
    .ffn_up.weight                      [2048,5632]     Q4_0
blk.1
    .attn_k.weight                      [2048,256]      Q4_0
    .attn_norm.weight                   [2048]          F32
    .attn_output.weight                 [2048,2048]     Q4_0
    .attn_q.weight                      [2048,2048]     Q4_0
    .attn_v.weight                      [2048,256]      Q4_0
    .ffn_down.weight                    [5632,2048]     Q4_0
    .ffn_gate.weight                    [2048,5632]     Q4_0
    .ffn_norm.weight                    [2048]          F32
    .ffn_up.weight                      [2048,5632]     Q4_0
...
blk.21
    .attn_k.weight                      [2048,256]      Q4_0
    .attn_norm.weight                   [2048]          F32
    .attn_output.weight                 [2048,2048]     Q4_0
    .attn_q.weight                      [2048,2048]     Q4_0
    .attn_v.weight                      [2048,256]      Q4_0
    .ffn_down.weight                    [5632,2048]     Q4_0
    .ffn_gate.weight                    [2048,5632]     Q4_0
    .ffn_norm.weight                    [2048]          F32
    .ffn_up.weight                      [2048,5632]     Q4_0
output.weight                           [2048,32000]    Q6_K
output_norm.weight                      [2048]          F32
token_embd.weight                       [2048,32000]    Q4_0
```

If the model.gguf doesn't exist or is not a valid GGUF file (verified by checking the magic number), you only need to print the first line:

```
$ ./hw0205
GGUF: false
```

You can find more GGUF example files on HuggingFace. Additionally, you can use the tensor viewer to check the correctness of your work. Here is an example for the above-mentioned TinyLLaMa model.

Good Luck!

PS: The source of the structures used in this problem is in the following link.

https://github.com/ggerganov/ggml/blob/f5c9599cdba3133da0158dce061b33413b49f6fd/src/ggml.c#L20262-L20366

Follow up: Some models are VERY large (>20GB per file), can your program read them without OOM?

6 Bonus: Where is errno (5 pts)

In this class, I have shown you how error number works. As you can see, this is an **extern** variable. Please answer the following questions.

1. Where is **errno**?
2. Please find all **known** error numbers. Do not search them from Google but find them from **codes**.
3. Please define a new error number and use **perror** to show it.