Mini project

Design a program with suitable functions and data structures to encode and decode Morse code from and to readable text.

1. Background:

Morse code is a method often used together with an electrical telegraph to transfer text message remotely in late 19th century until early 20th century. People also send message encoded with Morse code by flashing light or waving flag. The Morse code encodes text characters as sequences of two different signal durations, call *dots* and *dashes*, or *dits* and *dahs*, i.e. "." and "-". The *dots* represent short signal and the *dashes* represent the long one. Reference of Morse code chart can be found at (select English version): https://www.itu.int/rec/R-REC-M.1677-1-200910-I/.

2. Problem:

Students are required to write a program in C/C++ that should open a file and figure out if it contains text data (English) or Morse code. It should then convert the file to another format, i.e. if the file contains text data, the program should convert the normal text into Morse code and save the result in another file.

3. Technical requirements:

The program should have the following features:

a) It should be run as command-line, e.g.:
   - `C:\morse <inputfile> <outpufile>`

   Input file and output file can have different name

b) A few optional commands can be:
   - –h : print out the help to instruct user of the usage on the screen and save the instruction to a file name readme.txt, override if readme.txt has already existed.
   - –t : force to take the input file as a text file and the output file as the Morse code one.
   - –m : force to take the input file as Morse code and the output file as the interpreted text file.
   - –c : print the statistics of the successful conversion to the screen including:
     - Input file name and output file name
     - Time when the conversion completed (formatted as YYYY-MM-DD HH:MM:SS)
     - Duration of the conversion
     - Word count in input file
     - Number of word have been converted successfully
     - Number of work contained errors
     - Total number of characters in input file
     - Number of characters have been converted
     - Number of characters cannot be converted

   This information should also be written in a log file in text format which is named as <inputfile>_<outputfile>_<date>_<time>.log. <inputfile> and <outputfile> are input and output file names without their extension; <date>_<time> is the date (YYYYMMDD) and time (HHMMSS) when the conversion is completed. Eg.: input_output_20211115_101005.log

E.g.:

```
Input file: inputfile.txt
Output file: outputfile.txt
Time complete: 2021-Nov-01 14:00:00
Duration [seconds]: 10.0
Word count in input file: 254
Word converted: 254
Word with errors: 0
Total number of characters: 1234
Number of characters have been converted: 1234
Number of characters are NOT converted: 0
```

Notes on optional arguments and the command-line:

- If user enter an unrecognized command, the program should exit and print out an error message "Error XX: Unknown command. Type "morse –h" for help" where XX is the error code students define.
- "–t" and "–m" cannot be used together in the same command. If both occurred in the command, the command is considered unknown. In case both "–t" and "–m" are not in the commands, the program should detect the file format as described in c) and convert automatically.
- If there is no argument is provided, the program should print to screen "Error XX: Unrecognized command. Type "morse –h" for help" and exit.
- Do not allow more than 4 arguments in the same command.
- Any other issues related to command-line arguments, e.g. missing arguments, changing order of appearance, etc. should also be taken care.

Example of the commands with optional argument:

- `C:\morse <inputfile> <outpufile> -t`

  The above command treats the input file as a text file and convert it to a Morse code file which is the output.

c) The program needs to verify the format of the input file before converting its content as following:
   - If the input file contains only "." (dot), "-" (dash), "/" (forward slash) and white space, it should be Morse code file. If any other character appears in the file, it should be text file.
   - Characters in Morse code are separated by space
   - Words in Morse code are separated by "/"
   - Words in text file are separated by space.
   - Each line in file is ended with "\n".
   - There is no difference between upper case and lower case characters when using Morse code. Therefor output text file can have all the characters in lower case.
   - A line can have (near) infinite length. Each line in input file should be converted into a line in output file respectively.

d) The program should handle any errors which can happen by showing the suitable and clear message with an error code. Student can define the error code yourself. A few errors can be listed here:
   - The program cannot open the input file. Message can be: "Error XX: FILENAME could not be opened", where XX is self-defined error code, FILENAME is the name of the file.
   - If output file has been existed, the program should ask if user want to override it or not: "Warning: FILENAME already exists. Do you wish to overwrite (y, n)?". The answer is "y" the program proceeds and overwritten the file, otherwise it exits. If the file with the same name is not allowed to access, an error message should be shown.
   - If the text file contains characters which do not have the corresponding ones in Morse code chart, the characters should be converted into "#" and an error message: "Error AB:

Unrecognized character C on line XX." should be shown with "AB" is the error code, "C" is the character that cannot be converted and "XX" is the line that character appears.

- If the Morse code file contains the wrong format codes (e.g. total number of "." and "-" is larger than 7), each wrong code should be converted to # in text. If Morse code is in the right format but unrecognized, i.e. it does not have the corresponding letter character, it should be converted to "*" in the output file. In both the cases, the error message like: "Error AB: Invalid Morse code CODE on line XX." should be shown where "CODE" is the Morse code that cannot be converted.
- All the error messages should be displayed on the screen. They should also be written in the log file mentioned in b); in case "–c" is used, the errors should be appended after the statistics information. Each messages should be in a different line.

4. Morse code chart:

a) Letters

	a	.–	i	..	r	.–.
	b	–...	j	.–––	s	...
	c	–.–.	k	–.–	t	–
	d	–..	l	.–..	u	..–
	e	.	m	––	v	...–
accented	e	..–..	n	–.	w	.––
	f	..–.	o	–––	x	–..–
	g	––.	p	.––.	y	–.––
	h	....	q	––.–	z	––..

b) Figures

1	.————	6	–....
2	..–––	7	––...
3	...––	8	–––..
4	....–	9	————.
5	.....	0	—————

c) Punctuation marks and miscellaneous signs

Full stop (period)	[.]	.–.–.–
Comma	[,]	––..––
Colon or division sign	[:]	–––...
Question mark (note of interrogation or request for repetition of a transmission not understood)	[?]	..––..
Apostrophe	[']	.————.
Hyphen or dash or subtraction sign	[–]	–....–
Fraction bar or division sign	[/]	–..–.
Left-hand bracket (parenthesis)	[(]	–.––.
Right-hand bracket (parenthesis)	[)]	–.––.–
Inverted commas (quotation marks) (before and after the words)	[" "]	.–..–.
Double hyphen	[=]	–...–
Understood		...–.
Error (eight dots)		.....
Cross or addition sign	[+]	.–.–.
Invitation to transmit		–.–
Wait		.–...
End of work		...–.–
Starting signal (to precede every transmission)		–.–.–
Multiplication sign	[×]	–..–
Commercial at[1]	[@]	.––.–.

Students can refer here for more information: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1677-1-200910-I!!PDF-E.pdf

5. Examples

Assuming that we have an input file: myinput.dat and an output file convert.txt. We can type as command-line interface:

```
$ morse myinput.dat conver.txt
```

a) Example 1: Normal conversion (no error)

The following is a example of Morse code used in 1844:

The original message:

What hath God wrought

Is translated into:

.-- .... .- - / .... .- - .... / --. --- -.. / .-- .-. --- ..- --. .... –

The above Morse code can be converted as:

what hath god wrought

b) Example 2: Error occurs:

The below codes:

-.-. ..... .- -/--- -. /.- / -- .- - ..-... .-.-.-

-... ..- .-.-.-.- --. / .. -. / .-/ .-. ..- --. .-.-.-

Are converted into:

c#at on a mat*.

bu*g in a rug.

Message errors:

Error 02: Invalid Morse code ..-... on line 1.

Error 02: Invalid Morse code .-.-.-.- on line 2.

6. Design:

Students can use top-down approach to design the program, i.e. it is required to provide a diagram to illustrate the functions developed and their relationship. If the program is written in multiple files, a diagram need to be provided to illustrate the relationship of the files as well. What the structure is and how the files are designed and organized should also be introduced.

Flowchart diagrams should be provided for important / main functions. Analysis of the features, inputs and outputs of these functions should also be presented.

Important data structures used in the program should also be explained.

7. Coding style:

The coding style can follow GNU as described in this url: https://www.gnu.org/prep/standards/html_node/Writing-C.html

In short, the code writing should be formatted well by using proper curly bracket, indentation, line break to differentiate code blocks. Comments should be given to explain the code clearly for the readers. Variables and functions should be named properly in English, they should be concise and self-described. Hard-coding should be avoided. Students should not use any non-standard libraries.

8. Report and Submission guidelines:

- Students do the assignment as a team of 2-3 people.
- The assignment should be organized as one project folder.

- Students have to write a report of maximum 6 A4 pages (do not include source codes in it) in English using IEEE word template to explain:
  - The idea, design and implementation of the program.
  - Discussion and evaluation of the results.
  - What have been done and have not been done in your work.
  - Task allocation amongst the team (describe in a table).
  - Any further suggestions.
- The write-up should be structured and presented well as instructed in the template. The template file is attached together in Team assignment.
- Information from any external sources used in the report must be cited and the source must be listed in the References section.
- Students have to compress all the files including source codes, compile codes, input and output files, log files, report (in Word) file in one zip file (do NOT use *.rar or other compression format) and name it as "group_NUMBER_miniproject_20211.zip", e.g.: "group_1_miniproject_20211.zip" and upload on this class teams. Do not use any subfolders. If wrong name is submitted, 0.5 points will be deducted.

9. Evaluation:

- Assignment will be marks as below (total 10 points):
  - Complete program with all the requirements fulfilled creatively: 4 points
  - Good design and coding style: 3 points
  - Well written report in the correct format: 3 points
  - Late submission (during 1 week after the due date): -2 points
- Students must do the assignment yourselves.
  - It is strictly prohibited to copy source code / report from other students / groups or other sources.
  - Plagiarism from external sources will result in a consequence of at least 5 points deduction from the mid-term exam scores, submission with serious plagiarism will be given 0.
  - If two or more submissions from different groups are similar, all the groups will be given 0. It does not matter who copies from whom, thus students should keep the work confidential.
- Deduction of 2 points from mid-term score is also applied for no submission, if the offline mid-term exam can happen.
- If an offline mid-term exam cannot happen due to the Covid-19 pandemic; this assignment would be a part of the mid-term exams. In this case, no submission will result in mid-term score of 0.
- In case there is no offline mid-term exams, the contribution of this submission is in the final scores of the mid-term exams will be decided later.
- The deadlines (due date and closing date) are specified in Teams Assignment. Submission after the closing date is not accepted.
- Bonus points may be given in the final exams based on the scores of the assignment with the condition that the score without bonus must be greater than 4 (over 10):
  - 2 group with highest scores and earliest submission: 3 points on averages
  - The next 3 groups with high scores and earliest submission: 2 points on averages
  - The next 4 groups with high scores and earliest submission: 1 points on averages