Project 1 - Morse Code Converter

CS 0447 — Computer Organization & Assembly Language

Check the Due Date on the CourseWeb

The purpose of this project is for you to practice writing assembly language by implementing a Morse code converter that can convert a string of letters into a Morse code and vice versa.

Introduction to Morse Code

Morse code (Wikipedia) is a method of transmitting **text** information as a series of on-off tones that can be directly understood by a skilled listener or observer without special equipment. Each Morse code symbol represents either a text character is represented by a unique sequence of dots (`.`) and dashes (`-`). Each dot or dash is followed by a short silence, equal to to dot duration. The letters of a word are separated by a space equal to three dots (one dash), and the words are separated by a space equal to seven dots. The chart of the Morse letters is shown below:

Letter	Morse Code	Letter	Morse Code
A	`.-`	N	`-.`
B	`-...`	O	`---`
C	`-.-.`	P	`.--.`
D	`-..`	Q	`--.-`
E	`.`	R	`.-.`
F	`..-.`	S	`...`
G	`--.`	T	`-`
H	`....`	U	`..-`
I	`..`	V	`...-`
J	`.---`	W	`.--`
K	`-.-`	X	`-..-`
L	`.-..`	Y	`-.--`
M	`--`	Z	`--..`

Note that lowercase and uppercase of the same letter are translated into the same Morse code.

Part I: Convert a String to a Morse Code (30 Points)

For this part, ask a user to enter a string. You can assume that the string the user enters will contain only letters (a – z, A – Z) and some spaces. No numerals or symbols will be entered by a user. Once user press the `[Enter]` key, your program should print out the Morse code associated with the string user entered. Note that you should add one space character in between two letters and two spaces between two words. The following is an example of an output:

```
Please enter a string: I am Groot
..   .- --   --. .-. --- --- -
```

Note that there are two spaces between `..` (I) and `.-` (a) and only one space between `.-` (a) and `--` (m). Do not forget about uppercase vs lowercase letter. They use the same Morse code.

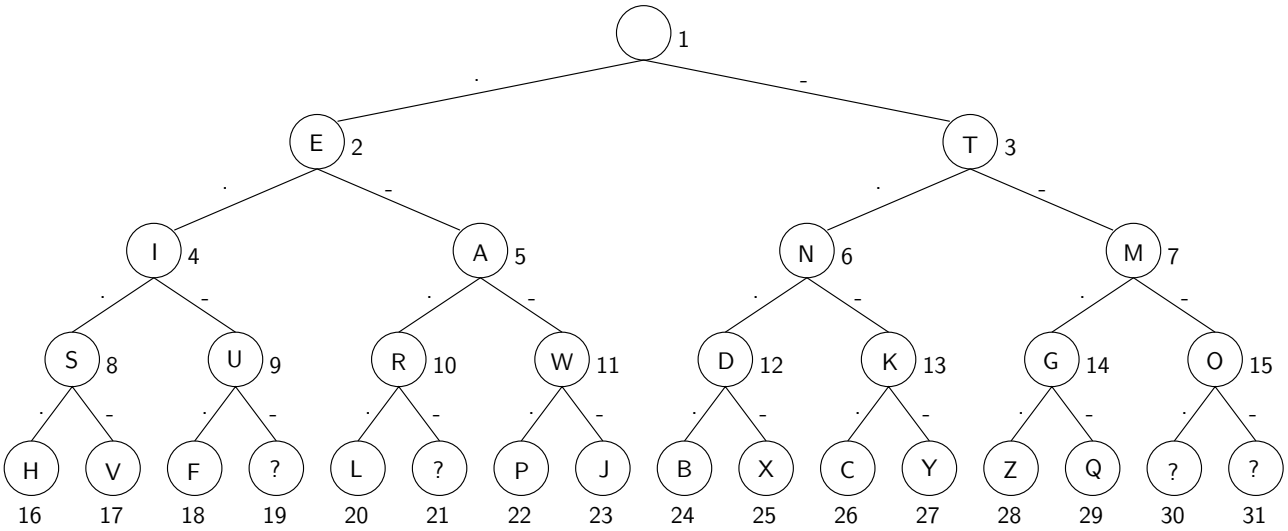
Part II: Convert a Morse Code to a String (40 Points)

For this part, ask a user to enter a string representation of a Morse code using dots (`.`), dashes (`-`), and spaces in the same format as the output from Part I (one space between letters and two spaces between words). Show the result in all caps as shown below:

```
Please enter a morse code: ..   .-.. --- ...- .   .- ... ... . -- -... .-.. -.--
I LOVE ASSEMBLY
```

For this part, you can assume that a user will not enter an invalid Morse code (e.g., `----`).

Convert from Morse code back to a string may seem like a complicate task. But it is actually a lot easier than you think. Consider the following binary tree:



This tree is carefully crafted according to the Morse code. You can use this tree to decode a Morse code back to a character. The process is pretty straightforward as follows:

1. Start at the top of the tree.
2. If you see a dot, go down one level to the left but if you see a dash, go down one level to the right.
3. Keep doing this until you reach the end of the code.

For example, if the Morse code is `.--.`, starting from the top go down left, right, right, and left. You will find yourself in the node containing P. Do not worry that I will ask you to implement a binary tree in assembly. That is the job of CS0445 for you to implement one but in Java. The

above binary tree can be captured by a very simple one dimensional array of characters. Note that there is a number next to each node. Imagine that a number is the index of the array which contains the character in that node. For example, the array at index 2 should contain E, at index 3 should contain T, at index 4 should contain I, and so on. To traverse this three down, first you initialize your `currentIndex` to 1. If you have to go down to the left, you simply update your current index by `currentIndex = currentIndex * 2`. If you have to go down to the right, you simply update your current index by `currentIndex = (currentIndex * 2) + 1`. Let's look at the same example `.--.`, here is a step-by-step in C style program:

```
currentIndex = 1;
// first one is a dot
currentIndex = currentIndex * 2;         // Now currentIndex is 2
// second one is a dash
currentIndex = (currentIndex * 2) + 1;   // Now currentIndex is 5
// third one is a dash
currentIndex = (currentIndex * 2) + 1;   // Now currentIndex is 11
// last one is a dot
currentIndex = currentIndex * 2;         // Now currentIndex is 22
```

According to the binary tree above, the index 22 should already contain the character P.

Part III: Put Them Together (30 Points)

For this part, you are going to put Parts I and II together into one program. First, your program should show the main menu asking a user whether he/she wants to convert a string to a Morse code, a Morse code to a string, or exit the program as shown below:

```
Main Menu
=========
  1. String to Morse code
  2. Morse code to string
  3. Exit program
What do you want to do? (1 or 3):
```

Note that if a user enter an invalid number (e.g., 0, -5, or 6), your program should notify the user and ask for a number again as shown below:

```
Main Menu
=========
  1. String to Morse code
  2. Morse code to string
  3. Exit program
What do you want to do? (1 or 3): 5
Invalid option
What do you want to do? (1 or 3):
```

If a user enter 1, you program simply perform the task from Part I. Once a string is converted and display, simply show the main menu again. If a user enter 2, perform the task from Part II and

show the main menu again. If a user enter 3, simply terminate the program. Here is an example of a run:

```
Main Menu
=========
  1. String to Morse code
  2. Morse code to string
  3. Exit program
What do you want to do? (1 or 3): 1
Please enter a string: I am Groot
..   .- --   --. .-. --- --- -
Main Menu
=========
  1. String to Morse code
  2. Morse code to string
  3. Exit program
What do you want to do? (1 or 3): 5
Invalid option
What do you want to do? (1 or 3): 2
Please enter a morse code: .--. .. - - ... -... ..- .-. --. ....
PITTSBURGH
Main Menu
=========
  1. String to Morse code
  2. Morse code to string
  3. Exit program
What do you want to do? (1 or 3): 3

-- program is finished running --
```

Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. You should submit the file `morse.asm` via CourseWeb.