MORSE CODE: Programming Tasks

The following tasks require you to open the skeleton program and make modifications.

Task 1

This task refers to `GetMenuOption`.

Currently, the program allows any single character to be entered as a choice from the menu. Modify `GetMenuOption` subroutine so that all values entered are converted to upper case and only valid choices can be made. If an invalid choice is entered, the user should be prompted with the message:

```
Invalid choice, please choose a letter from the menu:
```

This this should repeat until they have entered a valid choice. For example:

- Entering 'a' should result in the above prompt
- Then pressing Enter should make the same prompt appear again
- Finally entering 'S' should take you to the 'Send Morse code' option

Note that the first prompt to enter a choice from the menu should remain the same.

Evidence you need to provide: **(9 marks)**

- Your amended SOURCE CODE PROGRAM for `GetMenuOption`
- One screen capture showing the *menu choice*, the *prompt* and *result* for the following sequence:
  - Begin the program and enter 'y' at the prompt
  - Press Enter at the prompt
  - Enter 'SS' at the prompt
  - Enter 'R' at the prompt
- One screen capture showing the *menu choice*, the *prompts* and *result* for the following sequence:
  - Begin the program and enter 'x' at the prompt

Task 2

This task refers to `SendReceiveMessages` and `SendMorseCode`.

The program currently only accepts upper case letters. Modify the code so that full stops can be included using the sequence '`.-.-.-`' (dot, dash, dot, dash, dot dash):

- Alter the main code to add an additional constant called FULLSTOP (technically it's a variable in Python but we use the convention of uppercase to indicate a constant).
- Modify the `Letter` and `MorseCode` lists in `SendReceiveMessages` so that an *additional* element is added onto the end of each list for the full stop. Modify the `Dot` and `Dash` lists to ensure that a full stop can be correctly received in a transmission.
- Modify `SendMorseCode` so that a full stop is correctly identified (using the constant) and sent (using the 28th element of the `MorseCode` list).

Note that you will need to put the *`message2.txt`* file into the same folder as your program for this task.

Evidence you need to provide: **(8 marks)**

- Your amended SOURCE CODE PROGRAM snippet showing the additional constant
- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your amended SOURCE CODE PROGRAM for `SendMorseCode`
- One screen capture showing the *values entered* and the *result* for the following sequence:
  - Run the program and enter 'S' at the prompt
  - Enter 'S.O.S.' at the prompt
- One screen capture showing the *result* for the following sequence:
  - Run the program and enter 'R' at the prompt
  - Enter 'message2.txt' at the prompt

Task 3

This task refers to `DisplayMenu` and `SendReceiveMessages`. It also involves the creation of a new subroutine `PrintMorseCodeSymbols` which will have two parameters, the lists `Letter` and `MorseCode` from `SendReceiveMessages`.

Modify `DisplayMenu` and `SendReceiveMessages` to add the following as the third menu option (before X):

```
    P – Print Morse code symbols
```

This new menu option will need to call a new subroutine `PrintMorseCodeSymbols` and pass two arguments, the lists `Letter` and `MorseCode`. The subroutine should print out a table of all the Morse code letters and symbols in the following format:

```
Letter | Symbol
  A    |  .-
  B    |  -...
  C    |  -.-.
```

Evidence you need to provide: **(9 marks)**

- Your amended SOURCE CODE PROGRAM for `DisplayMenu`
- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your new SOURCE CODE PROGRAM for `PrintMorseCodeSymbols`
- One screen capture showing the main menu, selection of option P and the FULL table of symbols

Task 4

This task refers to `DisplayMenu, SendMorseCode` and `SendReceiveMessages`. It also involves the creation of a new subroutine `TransmitMorseCode` which will have one parameter, the list `MorseCode` from `SendReceiveMessages`.

Modify `DisplayMenu` and `SendReceiveMessages` to add the following as the third menu option (before X):

```
    T – Transmit Morse code
```

This new menu option will need to call a new subroutine, `TransmitMorseCode` and pass one argument, the list `MorseCode`. The new subroutine should call the existing subroutine `SendMorseCode` which will need to be modified to return the message instead of printing it out. (Note that you will also need to modify `SendReceiveMessages` to print out the return value instead of just calling it.) The new subroutine should then ask you for a file name and convert the Morse code message to transmission signals and store it in the file.

For example:

- The user selects option 'T' from the menu and is asked to enter their message
- They enter 'TEA TIME'
- The program prompts them for a file name and they enter 'message4.txt'
- The program generates the transmission (===   =   = ===   ===   = =   === ===   =   ) and stores it in the file `message4.txt`

Evidence you need to provide: **(21 marks)**

- Your amended SOURCE CODE PROGRAM for `DisplayMenu`
- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your new SOURCE CODE PROGRAM for `TransmitMorseCode`
- One screen capture showing the following sequence:
    - Run the program and enter 'T' from the main menu
    - Enter the message: 'ZIG ZAG'
    - Enter 'message4.txt' as the file name
    - Select option R from the main menu
    - Enter the file name 'message4.txt'

Task 5

This task refers to `SendReceiveMessages`, `ReceiveMorseCode` and `Decode`.

Currently, if an invalid sequence of dots and dashes is received, the program will return an incorrect letter instead of presenting a suitable error message.

Modify `SendReceiveMessages` to pass the list of valid symbols as the (new) fourth argument to `ReceiveMorseCode` and modify `ReceiveMorseCode` to pass the list of valid symbols as the (new) fifth argument to `Decode`.

You should decode an invalid character(s) as the asterisk (*) symbol and print out an error message stating the invalid sequence of dots and dashes that was received. You will use the message5.txt file to test this.

For example:

```
Enter your choice: R
Enter file name: message5.txt
*     Invalid Symbol (-.---) received.     *
 -.--- -
*T
```

Evidence you need to provide: **(9 marks)**

- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your amended SOURCE CODE PROGRAM for `ReceiveMorseCode`
- Your amended SOURCE CODE PROGRAM for `Decode`
- One screen capture showing choosing option R from the main menu and entering a file name of 'message5.txt'. (Note you will need to put message5.txt in the same folder as the program)

Task 6

This task refers to `GetTransmission`.

The program currently expects the full file name to be typed in (including the .txt extension), but it would be better if this was flexible.

Modify the `GetTransmission` subroutine so that it functions properly, with or without the filename extension.

No changes should be made to any of the prompts.

Evidence you need to provide: **(4 marks)**

- Your amended SOURCE CODE PROGRAM for `GetTransmission`
- One screen capture showing choosing option R from the main menu and entering a file name of 'message6'
- One screen capture showing choosing option R from the main menu and entering a file name of 'message6.txt'

Task 7

This task refers to `DisplayMenu` and `SendReceiveMessages` and involves creating a new subroutine `ConvertMorseCode`.

Currently, there is no option for the message to be entered in Morse code.

Modify `DisplayMenu` and `SendReceiveMessages` to add the following as the third menu option (before X):

```
C – Convert Morse code
```

This new menu option will need to call a new subroutine `ConvertMorseCode` and pass two arguments, the lists `MorseCode` and `Letter`. The new subroutine should ask the user to enter the message in Morse code and print out the decoded message. It should accept only valid Morse code and print out an error message if any symbol is invalid.

Evidence you need to provide: **(17 marks)**

- Your amended SOURCE CODE PROGRAM for `DisplayMenu`
- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your new SOURCE CODE PROGRAM for `ConvertMorseCode`
- One screen capture showing all of the input and output for the following sequence:
    - Run the program and enter ‘C’ from the main menu
    - Enter the Morse code: .... ..   - .... . .-. .
- One screen capture showing all of the input and output for the following sequence:
    - Run the program and enter ‘C’ from the main menu
    - Enter the Morse code: .... .-.--- .-.. .-.. ---

Task 8

This task refers to `SendMorseCode`.

Modify this subroutine to also generate the quaternary for the message to be sent. It should print this out after the encoded message in Morse code (on a separate line).

Quaternary Symbols:

- Letter separator (0)
- Word separator (1)
- Dot (2)
- Dash (3)

Encoding Examples:

- Three dots: 222
- Three dashes: 333
- The word ‘son’: 2220333032
- The phrase ‘is a’: 220222123

Evidence you need to provide: **(10 marks)**

- Your amended SOURCE CODE PROGRAM for `SendMorseCode`
- One screen capture showing all of the input and output for the following sequence:
    - Run the program and enter ‘S’ from the main menu
    - Enter the message: ‘TEST MSG’

Task 9

This task refers to `DisplayMenu` and `SendReceiveMessages`. It also involves the creation of a new subroutine `SendEncryptedMorseCode` which will have one parameter, `MorseCode`.

Modify `DisplayMenu` and `SendReceiveMessages` to add the following as the third menu option (before X):

```
E – Send encrypted message
```

This new menu option will need to call a new subroutine, `SendEncryptedMorseCode` and pass one argument, the list `MorseCode`. The new subroutine should ask the user to enter a message, then ask the user what the Caesar Cipher Shift for the message is. It should then apply the shift (but not shift spaces) and produce the Morse code for based on the cipher text for the message.

For example:

- User enters the message 'I AM'
- User chooses a Caesar Cipher Shift of 3
- Message is shifted by 3 to L DP (not displayed)
- Morse code version of the message is displayed: `.-..     -.. .--.`

Evidence you need to provide: **(16 marks)**

- Your amended SOURCE CODE PROGRAM for `DisplayMenu`
- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your new SOURCE CODE PROGRAM for `SendEncryptedMorseCode`
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'TEST MSG'
  - Enter a Caesar Cipher Shift of: '12'
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'TEST MSG'
  - Enter a Caesar Cipher Shift of: '-5'
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'TEST MSG'
  - Enter a Caesar Cipher Shift of: '50'

Task 10

This task refers to `SendMorseCode` and involves the creation of a new subroutine called `CalculateTransmissionTime` that will take one parameter (the message in Morse code) and return a single integer which represents the number of time units required to send the message.

Modify `SendMorseCode` so that it makes a call to `CalculateTransmissionTime` passing a variable containing the message in Morse code as the argument. It should retrieve the value returned and print it out in a suitable message of the following format:

Your message will take 80 time units to send.

Note: *When calculating the length of time in time units, a dot is 1 time unit and a dash is 3 time units. The gap between dots, dashes or spaces is 1 time unit. The gap between letters is 3 time units (1 from the final dot or dash and 2 additional time units to indicate the end of a letter). The gap between words (a space) is 7 time units (3 from the space at the end of a letter and 4 additional time units to indicate the end of a word).*

Evidence you need to provide: **(13 marks)**

- Your amended SOURCE CODE PROGRAM for `SendMorseCode`
- Your new SOURCE CODE PROGRAM for `CalculateTransmissionTime`
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'TEST MSG'

Task 11

This task refers to `SendMorseCode`.

Modify the subroutine so that the user can put in the message in any case (upper, lower or mixed case).

If the input includes at least one lower case letter then the subroutine should print the following message:

```
      Only uppercase letters can be used, your message has be converted to:
```

… followed by the message in uppercase.

Evidence you need to provide: **(6 marks)**

- Your amended SOURCE CODE PROGRAM for `SendMorseCode`
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'TEST MSG'
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'Test Message'
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'S' from the main menu
  - Enter the message: 'test msg'

Task 12

This task refers to `ReceiveMorseCode`.

Modify the subroutine so that is prints out a message showing how many symbols and characters have been received. Only dots and dashes count as symbols and only letters count as characters (ignore spaces).

For example:

- User selects 'R' from the menu and enters a file name containing a transmission signal
- 8 symbols received: `- . .-   -..-`
- Which represent 4 characters: `TEA X`

Evidence you need to provide: **(6 marks)**

- Your amended SOURCE CODE PROGRAM for `ReceiveMorseCode`
- One screen capture showing all of the input and output for the following sequence:
    - Run the program and enter 'R' from the main menu
    - Enter the file name: message12.txt

Task 13

This task refers to `SendReceiveMessages` and `DisplayMenu`.

The program currently uses the *International Morse Code* but needs to be updated to be able to switch between that and the *American Morse Code* system.

Modify the subroutine `DisplayMenu` so that the menu informs the user what standard of Morse code is being used. You will need to pass in a Boolean argument (`InternationalMorseCode`) – True by default – to specify either International (True) or American (False).

Create the following new menu option:

```
V – Change to American Morse code
```

Once this menu option has been chosen and American Morse code is being used, the menu option will change to:

```
V – Change to International Morse code
```

This new menu option should appear as the third menu option before X. For example:

```
Main Menu
=========
R - Receive Morse code
S - Send Morse code
V - Change to American Morse code
X - Exit program

System is currently using the International version of Morse code.
Enter your choice:
```

Note there is no need to actually change all of the symbols and mappings for this task but of course it would be followed through by actually changing the lists Dash, Dot and MorseCode were the entire change to be implemented.

Evidence you need to provide: **(17 marks)**

- Your amended SOURCE CODE PROGRAM for `SendReceiveMessages`
- Your amended SOURCE CODE PROGRAM for `DisplayMenu`
- One screen capture showing all of the input and output for the following sequence:
    - Run the program and enter 'V' from the main menu
    - Enter 'V' again from the main menu
    - Enter 'V' a third time from the main menu

Task 14

This task is an extension of Task 4 which you will need to have solved first in order to attempt this.

This task refers to `TransmitMorseCode`.

Modify your solution so that before it writes the transmission signals to the file, it checks to see if the file exists and asks the user whether they would like to overwrite the file or choose another file name instead.

For example:

```
Enter file name: message4.txt
File already exists, would you like to overwrite it (Y/N)?
```

Evidence you need to provide: **(11 marks)**

- Your amended SOURCE CODE PROGRAM for `TransmitMorseCode`
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'T' from the main menu
  - Enter the message: 'TEST MSG'
  - Enter the file name: 'message4.txt'
  - Choose 'N'
  - Enter the file name 'message14.txt'
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'T' from the main menu
  - Enter the message: 'TEST MESSAGE'
  - Enter the file name: 'message14.txt'
  - Choose 'Y'
  - Choose 'R' from the main menu
  - Enter the file name: 'message14.txt'

Task 15

This task refers to `GetTransmission.`

After the transmission has been received, display a message saying how many symbols are in the file and break this down into the number of units of a signal (=) and the number of units of no signal (space). Leading and trailing spaces should not be counted.

For example:

```
45 symbols received in transmission consisting of 30 signals and 15 breaks.
```

Evidence you need to provide: **(7 marks)**

- Your amended SOURCE CODE PROGRAM for `GetTransmission`
- One screen capture showing all of the input and output for the following sequence:
  - Run the program and enter 'R' from the main menu
  - Enter the file name: 'message.txt'