## MORSE CODE: Programming Tasks

**Suggested Solutions and Mark Scheme (Python3)**

The following are recommended solutions, and not an exhaustive list of all possible solutions to each task. The marking guidance should be used as a guide only. Discretion should be used in awarding credit where alternative solutions are given.

## Task 1 (max. 9 marks)

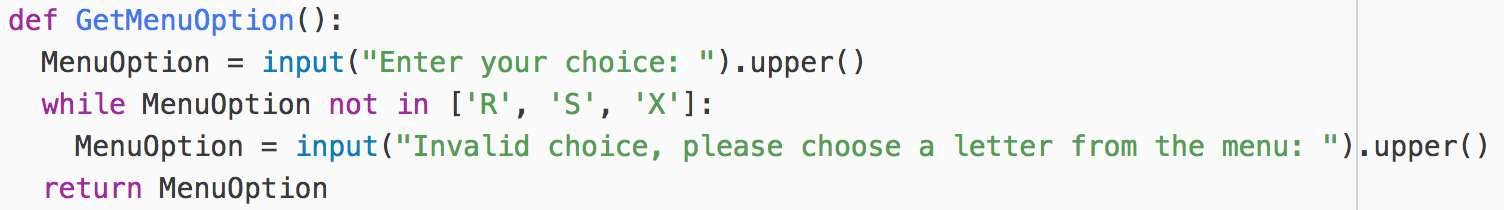
**1 mark** The user is always prompted with “Enter your choice: “ when the program is run

**1 mark** The input is converted to uppercase (or equivalent logic later such as checking both upper and lower case letters in the selection/iterative statement

**1 mark** There is an iterative statement that will continue to prompt the user when an invalid choice is made (even if it doesn’t work properly. Also accept an iterative statement with a flag variable and an appropriate selection statement inside the iterative statement which sets the flag variable correctly.

**1 mark** The condition for the iterative statement specifically prompts the user with “Invalid choice, please choose a letter from the menu: “ when anything other than “R”. “S” or “X” is entered (or the lowercase equivalents too if the input wasn’t converted to uppercase).

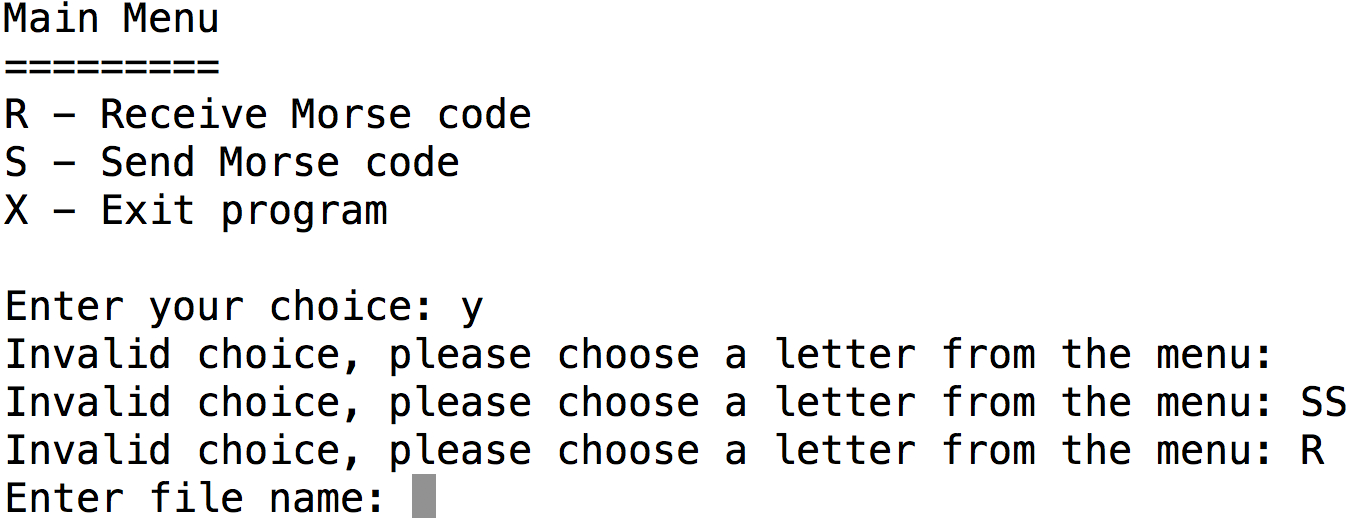
**1 mark** The input from the “Invalid choice” prompt is converted to uppercase (or equivalent logic later such as checking both upper and lower case letters in the selection/iterative statement



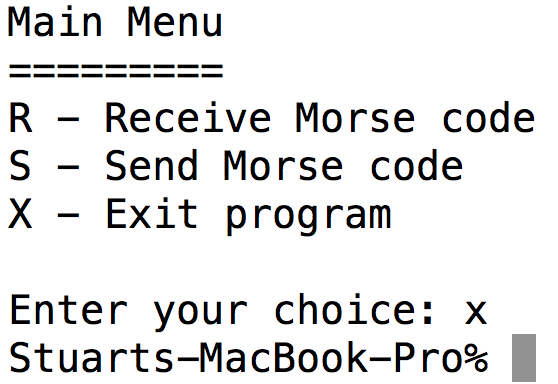
**1 mark** Screenshot shows ‘y’ was entered, resulting in the Invalid choice prompt

**1 mark** Screenshot shows nothing was entered (i.e. enter was pressed with no letter), resulting in the Invalid choice prompt, followed by ‘SS’ being entered at the prompt and another Invalid choice prompt

**1 mark** Screenshot shows ‘R’ was entered, resulting in the Enter file name: prompt



**1 mark** Screenshot shows ‘x’ was entered, followed by the program exiting:



## Task 2 (max. 8 marks)

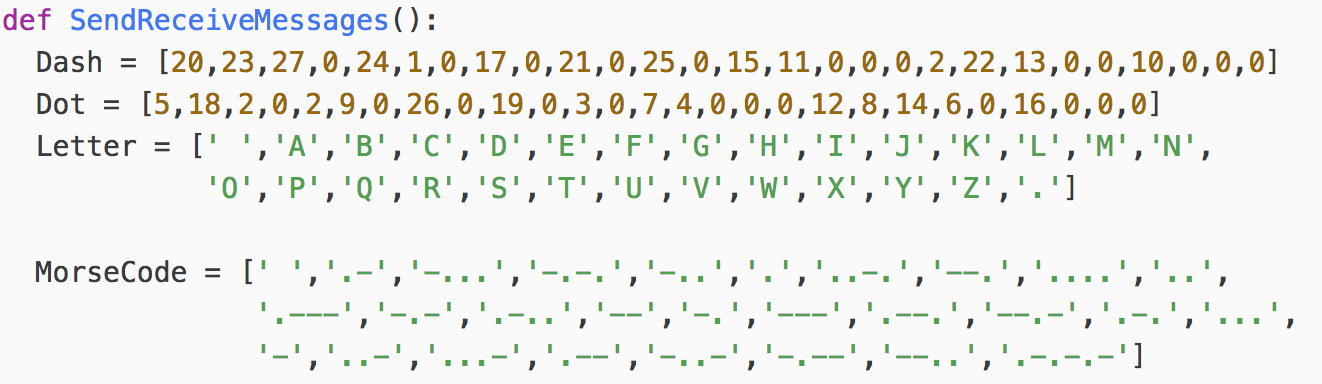
**1 mark** Addition of constant called FULLSTOP and set to the value ‘.’



**1 mark** Adding ‘.’ as the 28th element of the list Letter

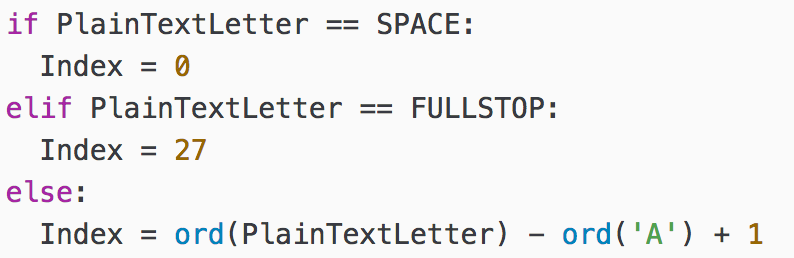
**1 mark** Adding ‘.-.-.-‘ as the 28th element of the list MorseCode

**1 mark** Modifying the lists Dot and Dash correctly so that a sequence of dot, dash, dot, dash, dot dash results in the number 27

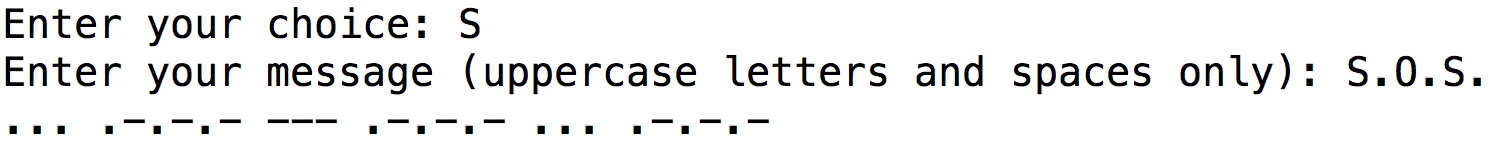


**1 mark** Modifying the selection statement in to correctly detect a full stop using the constant FULLSTOP

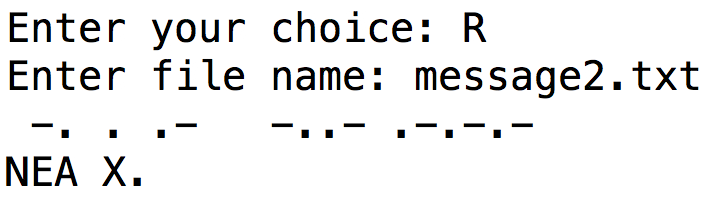
**1 mark** Selection statement correctly uses the sequence from the 28th element (numbered 27 as they start at 0)



**1 mark** Screenshot shows selecting S to send a message and entering S.O.S. as the message with the   
Morse Code being output as below:



**1 mark** Screenshot shows selecting R to receive a message and entering message2.txt as the filename and then the message is decoded as show below including the full stop.

****

## Task 3 (max. 9 marks)

**1 mark** Addition of new option to the menu by modifying DisplayMenu, only accept option exactly as given



**1 mark** Inclusion of menu option P in a new selection structure in SendReceiveMessages

**1 mark** New option calls the new subroutine PrintMorseCodeSymbols and passes only the arguments Letter and MorseCode

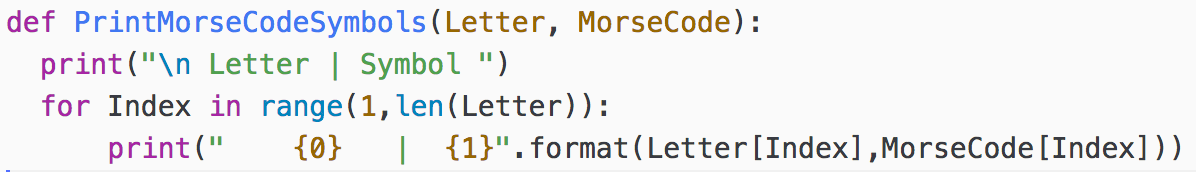


**1 mark** Code for subroutine PrintMorseCodeSymbols has two parameters (even if the parameters are not correctly named)

**1 mark** Print statement for the table heading outside of any iterative or selection structure

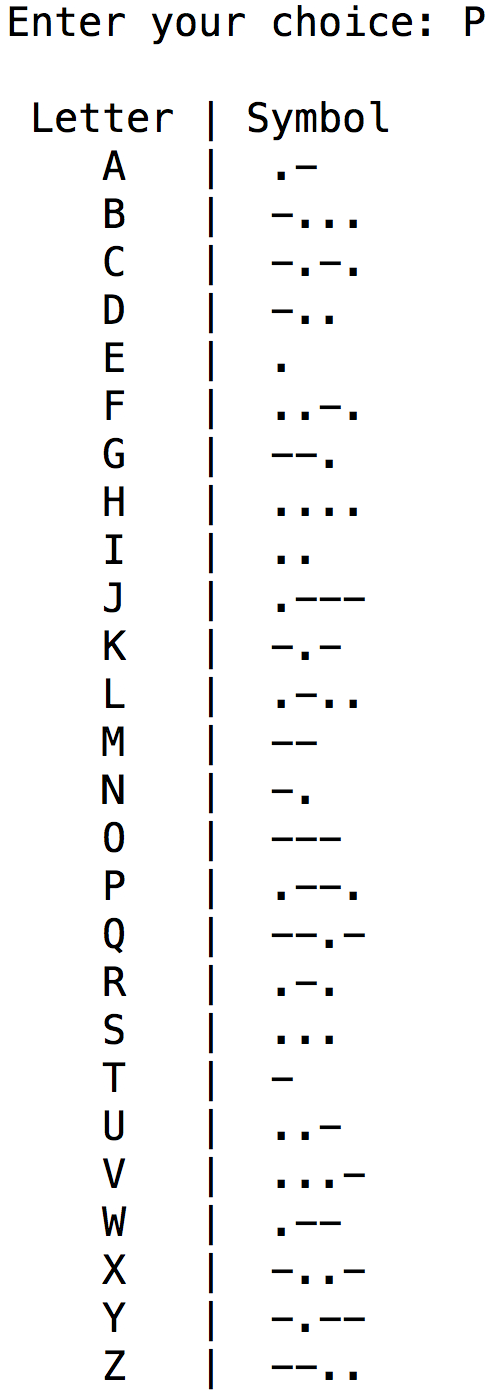
**1 mark** Iterative structure that will iterate through the Letter and MorseCode lists (reject hard coded number for the length)

**1 mark** Suitable code inside the iterative structure to print out a letter with its corresponding Morse code

****

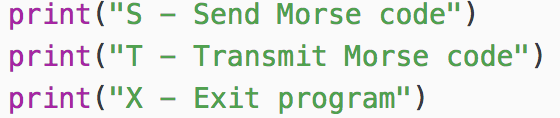
**1 mark** Screenshot shows two columns, one for Letter and another for Symbol, each of which contain all 26 letters and symbols correctly mapped to each other

**1 mark** Screenshot shows the table in precisely the correct format as per the question with the correct capitalisation and correct spacing (ignore failure to leave a blank line before printing the table)



## Task 4 (max. 21 marks)

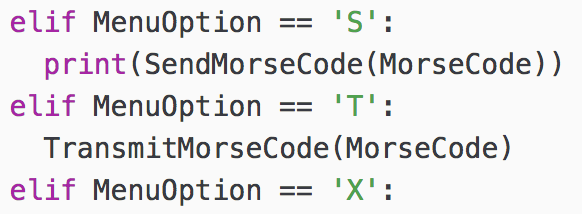
**1 mark** Addition of new option to the menu by modifying DisplayMenu, only accept option exactly as given



**1 mark** Inclusion of menu option T in a new selection structure in SendReceiveMessages

**1 mark** New option calls the new subroutine TransmitMorseCodeSymbols and passes only the argument MorseCode

**1 mark** Modification of menu option S to print out the result of the call to SendMorseCode



**1 mark** Code for subroutine TransmitMorseCode has one parameter (even if parameter not correctly named)

**1 mark** Call to SendMorseCode passing the argument of MorseCode (accept whatever the parameter was)

**1 mark** Result of call to SendMorseCode stored in a variable

**1 mark** Suitable iterative structure to go through the Morse code version of the message

**1 mark** Selection statement to store different transmission strings based on the Morse code symbol

**1 mark** Inclusion of space, dot and dash in the selection statement

**1 mark** Selection statement correctly handles putting a single space between symbols

**1 mark** Selection statement correctly handles putting a total of three spaces between letters

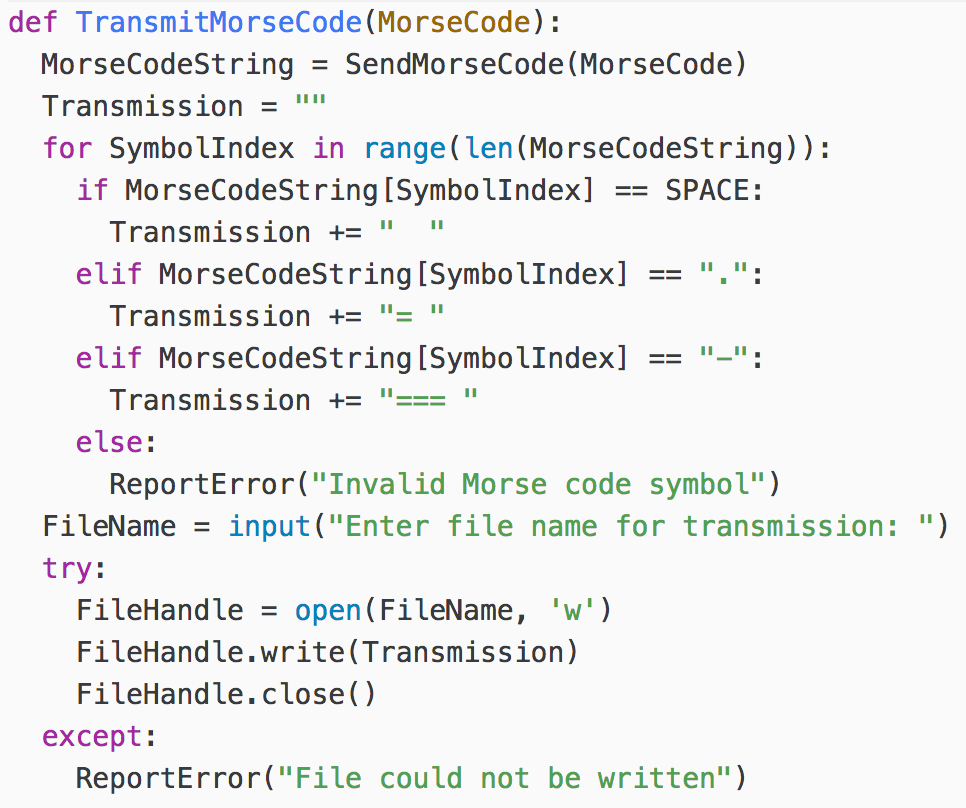
**1 mark** Selection statement correctly handles putting a total of seven spaces between words

**1 mark** Suitable prompt to enter a file name

**1 mark** Transmission string correctly written to the file

**1 mark** File is closed after being written to

**1 mark** Using a try…except… structure with an appropriate error message to protect the file handling

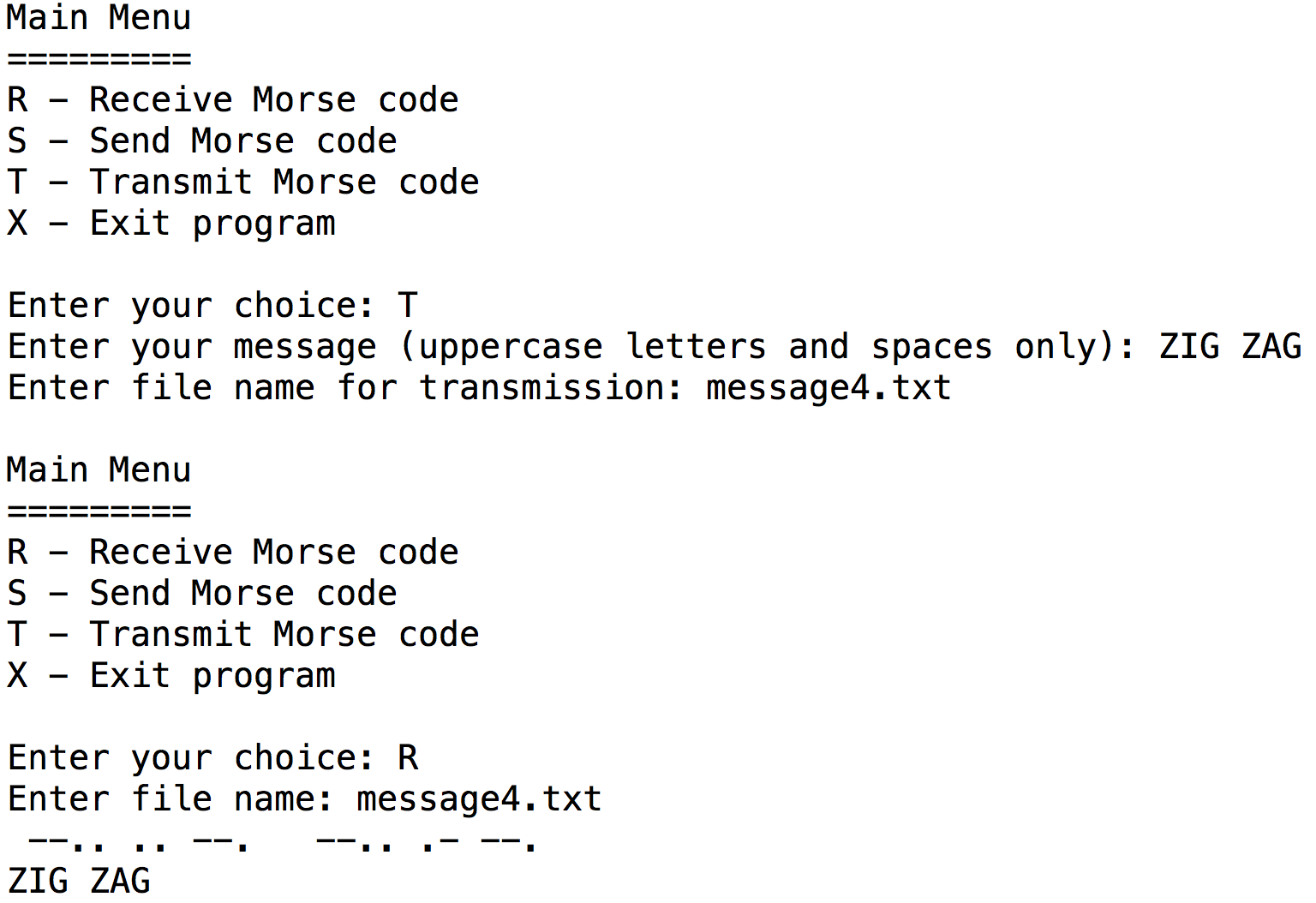
****

**1 mark** Screenshot shows choosing option T from the menu and entering the message ZIG ZAG (capital letters with a space between the two words)

**1 mark** Screenshot shows a prompt for the file name and the user entering message4.txt at the prompt

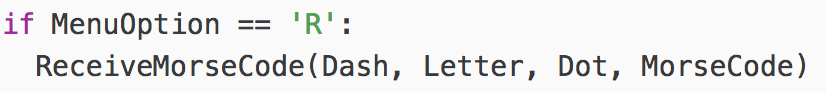
**1 mark** Screenshot shows the user choosing option R from the menu and entering the file name message4.txt at the prompt

**1 mark** Screenshot shows the correct Morse code and decoded message as per the screenshot below



## Task 5 (max. 9 marks)

**1 mark** Inclusion of new argument in the called to ReceiveMorseCode in SendReceiveMessages



**1 mark** Addition of new parameter to ReceiveMorseCode



**1 mark** Modification of the call to Decode in ReceiveMorseCode to specify an additional argument

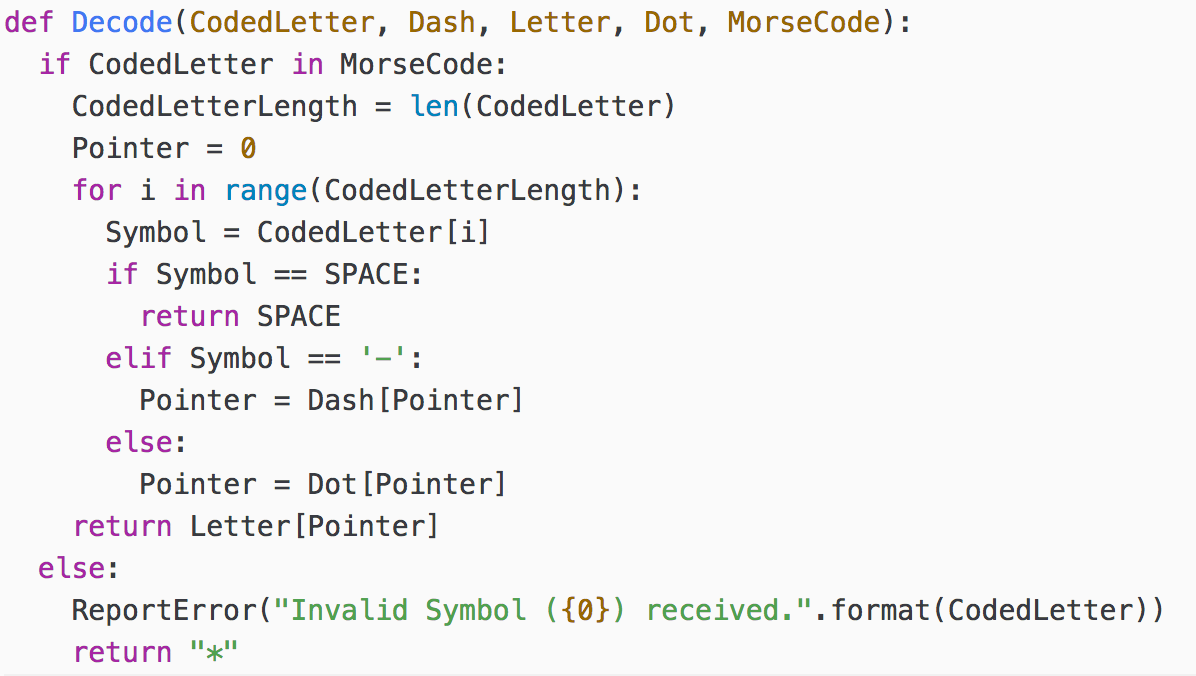


**1 mark** Addition of new parameter to Decode

**1 mark** Selection statement to include/exclude valid letters (depending on logic)

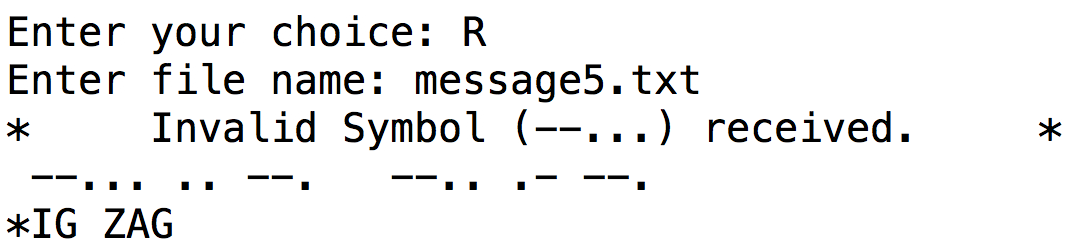
**1 mark** Error is reported if the CodedLetter doesn’t represent a valid sequence

**1 mark** An asterisk is returned if the sequence of dots and dashes is invalid



**1 mark** Screenshot shows an error message containing the invalid symbol (--…)

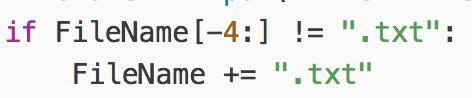
**1 mark** Screenshot shows the decoded message as \*IG ZAG



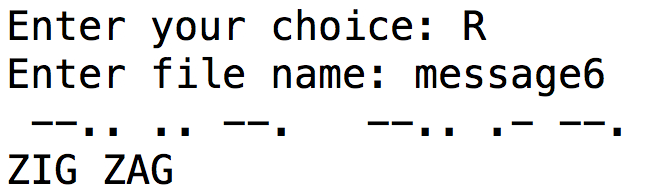
## Task 6 (max. 4 marks)

**1 mark** Selection statement to check if the last four characters of the FileName are .txt (accept any reasonable method of isolating and checking the last four characters only)

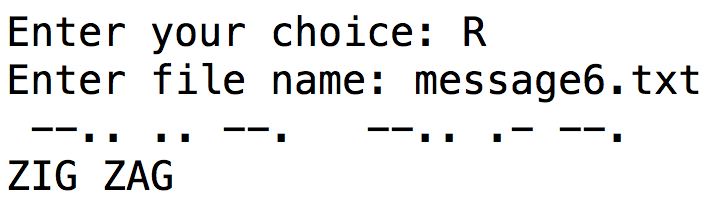
**1 mark** .txt correctly appended to the FileName if it is not already the last four characters (and only then)

****

**1 mark** Screenshot shows the filename entered as message6 without an extension and the message being received exactly as shown

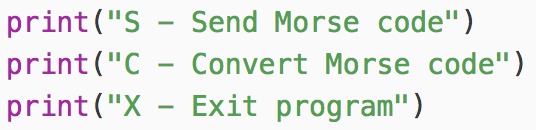
****

**1 mark** Screenshot shows the filename entered as message6.txt including the extension and the message being received exactly as shown



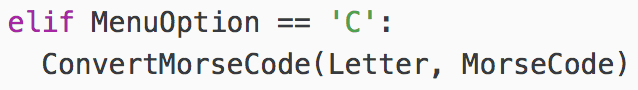
## Task 7 (max. 17 marks)

**1 mark** Addition of new option to the menu by modifying DisplayMenu, only accept option exactly as given



**1 mark** Inclusion of menu option C in a new selection structure in SendReceiveMessages

**1 mark** New option calls the new subroutine ConvertMorseCodeSymbols and passes two arguments Letter and MorseCode (accept them in either order)



**1 mark** Code for subroutine ConvertMorseCode has two parameters (even if parameters not correctly named)

**1 mark** User is asked to enter a message in Morse code

**1 mark** Input of Morse code from user stored in a variable with a meaningful identifier

**1 mark** Suitable iterative structure to go through the Morse code version of the message, symbol by symbol

**1 mark** Selection statement checks whether the symbol is a valid Morse code symbol

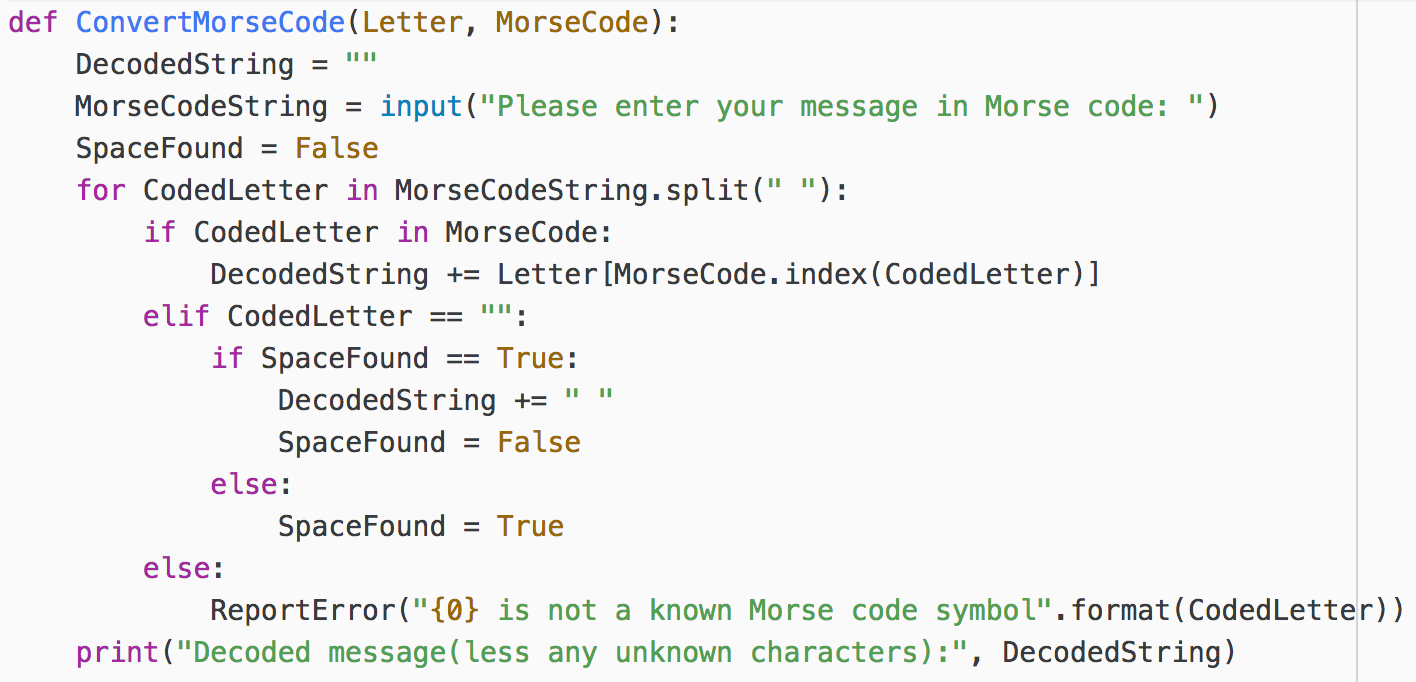
**1 mark** Inclusion of space in the selection statement

**1 mark** Selection statement correctly handles a single space between symbols and doesn’t print a space in the decoded message

**1 mark** Selection statement correctly handles a total of three spaces between letters and prints a single space in the decoded message

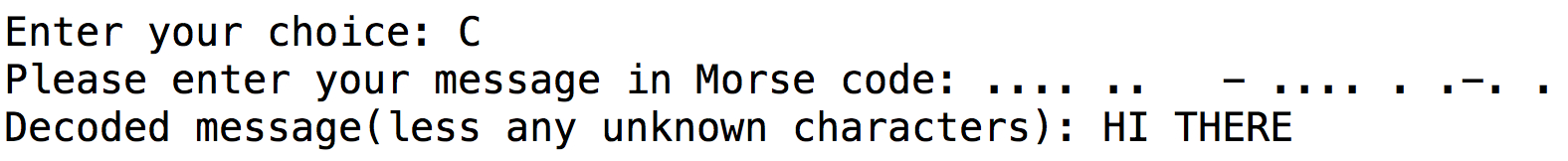
**1 mark** Printing out any invalid symbols received

*Accept alternative working solutions (at full marks) that call the subroutine Decode instead but penalise them one mark if it’s not modified to correctly detect any invalid symbols*



**1 mark** Screenshot shows choosing option C from the menu and entering the message:  
  
.... .. - .... . .-. .

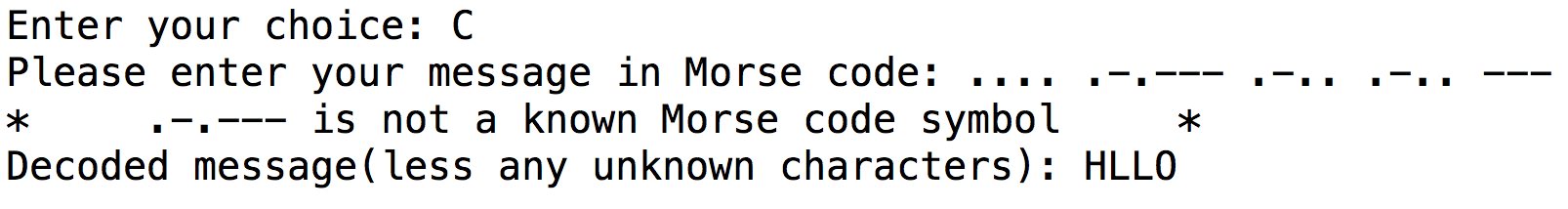
**1 mark** Screenshot shows the decoded message as: HI THERE



**1 mark** Screenshot shows choosing option C from the menu and entering the message:  
  
.... .-.--- .-.. .-.. ---

**1 mark** Screenshot shows the decoded message as: HLLO

**1 mark** Screenshot shows the symbol .-.--- as being invalid/not known



## Task 8 (max. 10 marks)

**1 mark** Suitable variable with meaningful identifier initialised to store the quaternary string

**1 mark** Selection statement to detect whether the letter is a space or a Morse code symbol

**1 mark** Selection statement placed inside appropriate iterative structure (which could be the existing one)

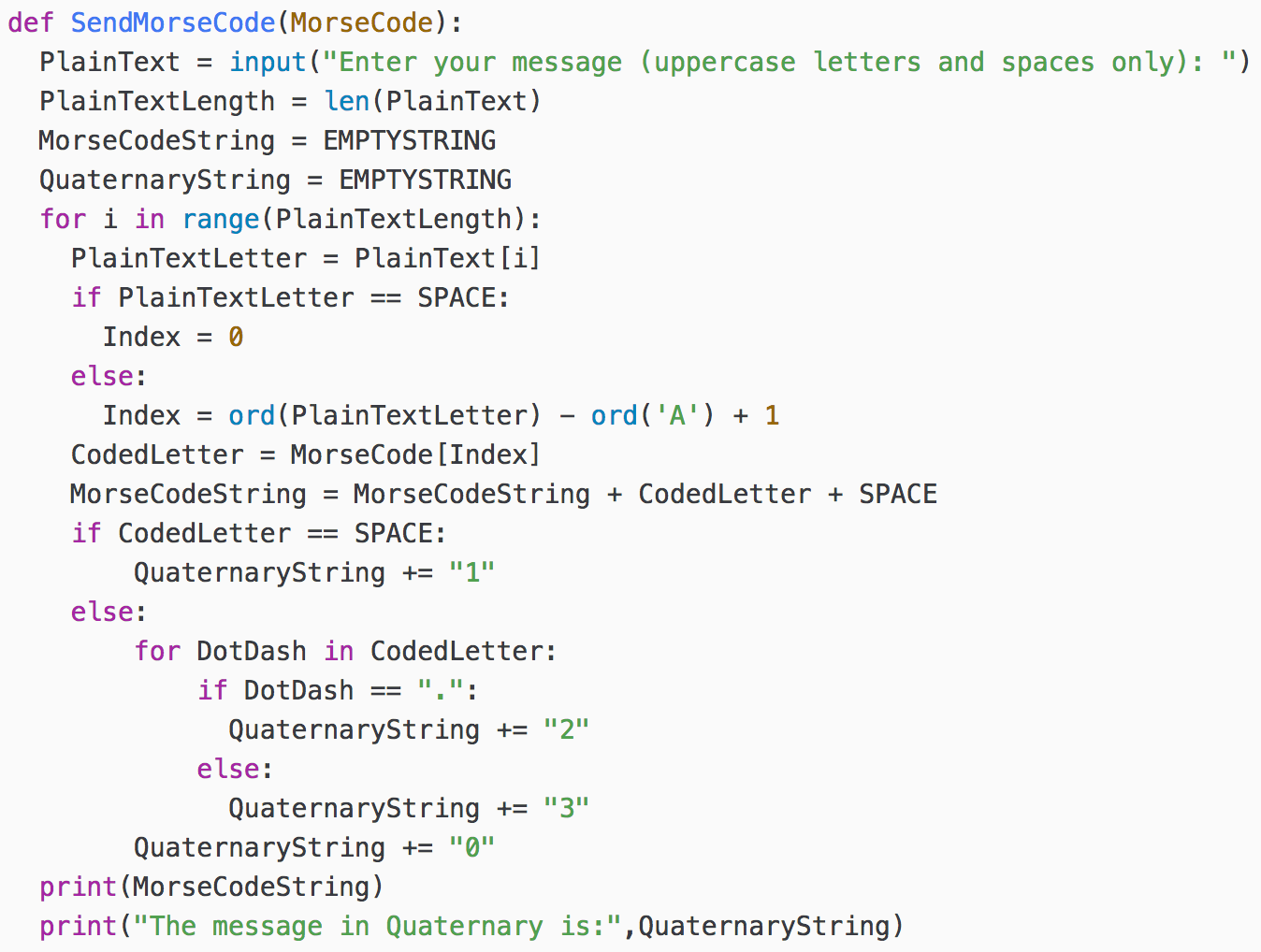
**1 mark** Selection statement correctly handles a space between words as 1 in quaternary

**1 mark** Selection statement contains an iterative statement to go through all of the dots and dashes in a Morse code symbol

**1 mark** Selection statement correctly handles a dot in a symbol as 2 in quaternary

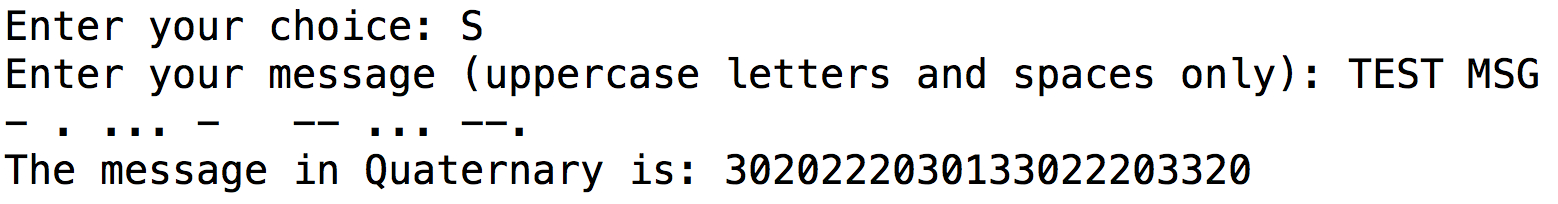
**1 mark** Selection statement correctly handles a dash in a symbol as 3 in quaternary

**1 mark** Selection statement correctly adds a 0 in quaternary after each complete Morse code symbol



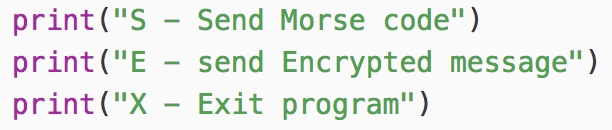
**1 mark** Screenshot shows S chosen from the main menu and the message entered correctly at the prompt

**1 mark** Screenshot shows the message correctly in quaternary AFTER the Morse code version of the message



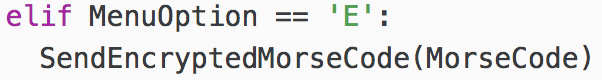
## Task 9 (max. 16 marks)

**1 mark** Addition of new option to the menu by modifying DisplayMenu, accept any capitalisation



**1 mark** Inclusion of menu option E in a new selection structure in SendReceiveMessages

**1 mark** New option calls the new subroutine SendEncryptedMorseCode and passes one argument MorseCode



**1 mark** Code for subroutine SendEncryptedMorseCode has one parameter (even not correctly named)

**1 mark** User is asked to enter a message in plain text which is stored in a variable with a meaningful identifier

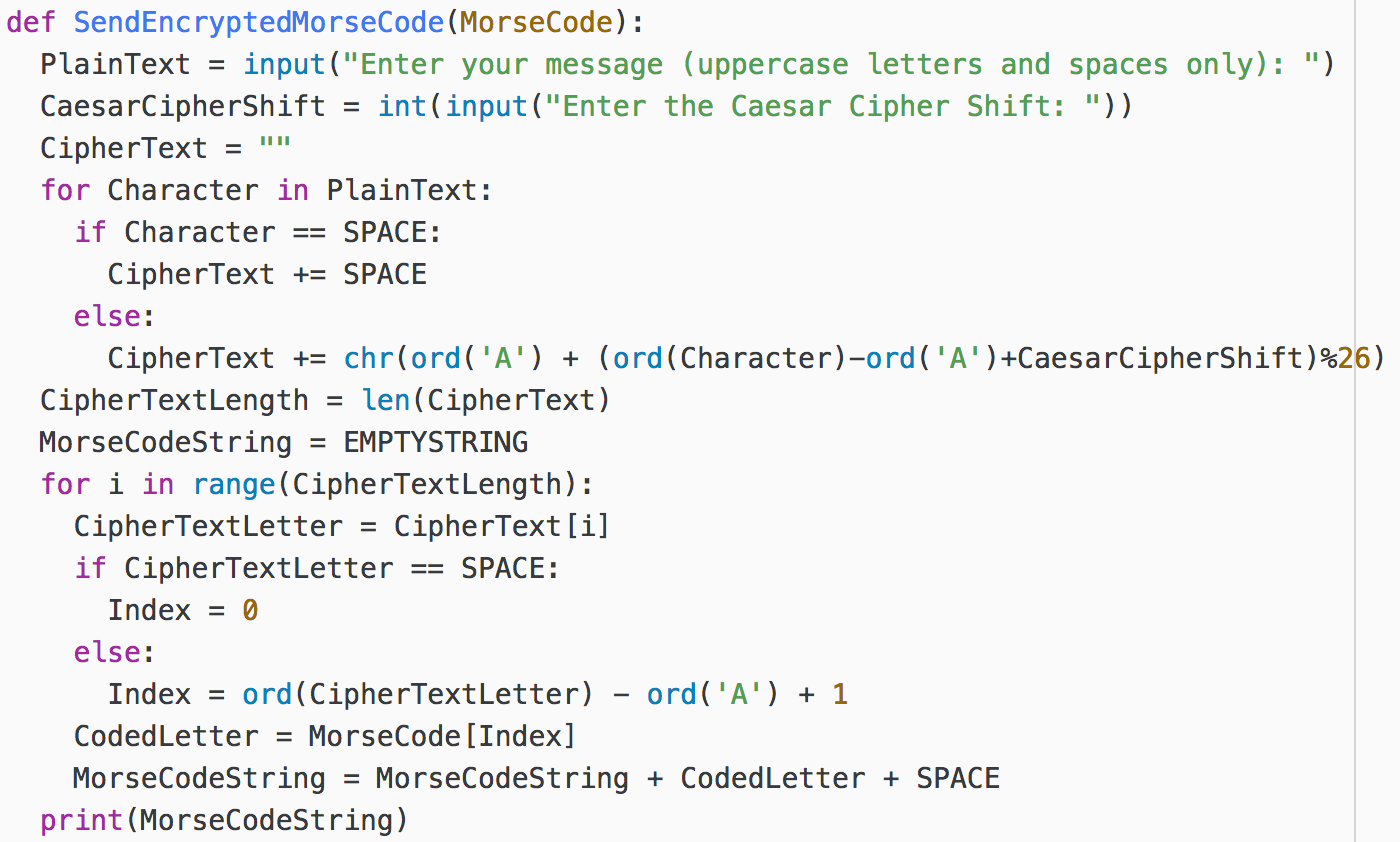
**1 mark** User is asked to enter a Caesar Cipher Shift which is converted to an integer and stored in a variable with a meaningful identifier

**1 mark** Iterative structure to go through the message entered, character by character

**1 mark** Selection statement inside the iterative structure that differentiates between spaces and non-spaces

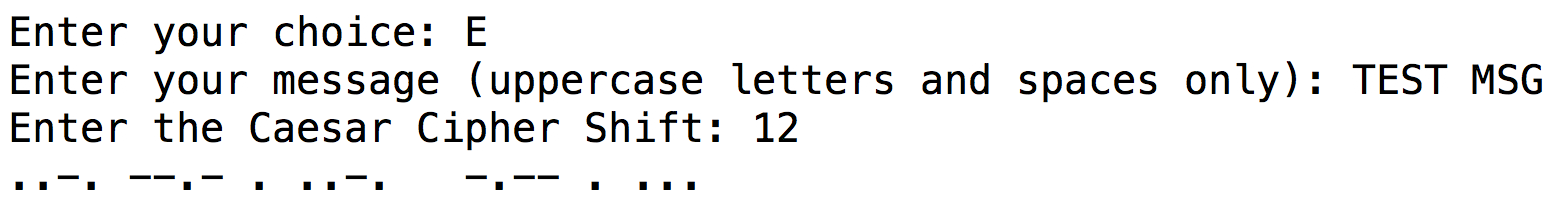
**1 mark** Character is correctly Caesar cipher shifted inside the selection statement (accept calls to library functions that do this for you). Do not award the mark if they fail to wrap the shift around (i.e. Z shifts to A)

**1 mark** Cipher text is then correctly converted to Morse code and printed out

****

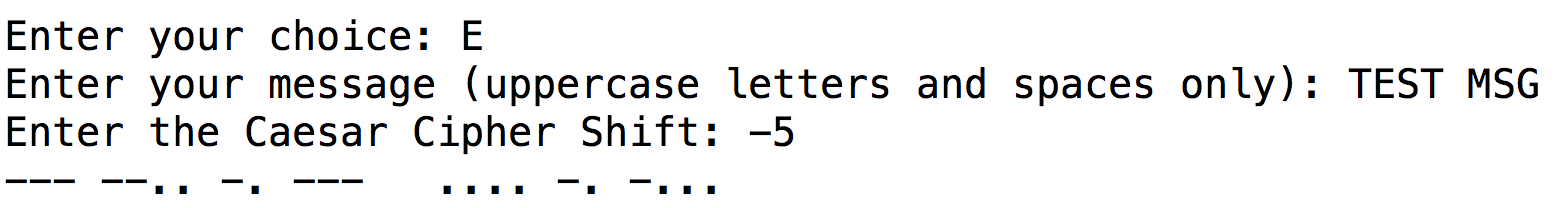
**1 mark** Screenshot shows choosing option E from the menu and entering the message TEST MSG and a Caesar Cipher Shift of 12

**1 mark** Screenshot shows the encoded message correctly



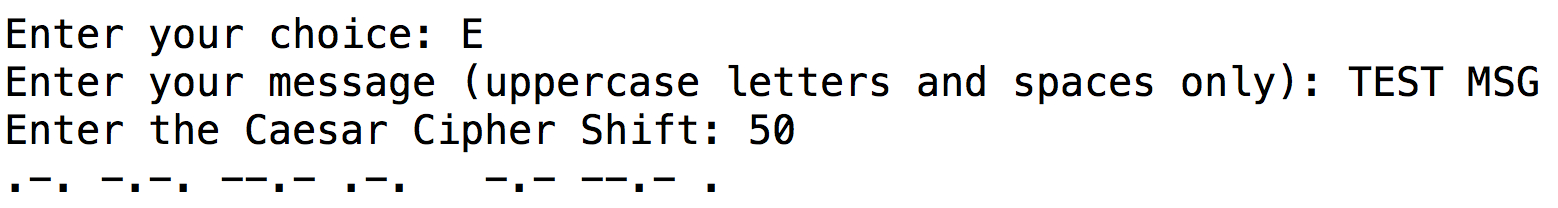
**1 mark** Screenshot shows choosing option E from the menu and entering the message TEST MSG and a Caesar Cipher Shift of -5

**1 mark** Screenshot shows the encoded message correctly



**1 mark** Screenshot shows choosing option E from the menu and entering the message TEST MSG and a Caesar Cipher Shift of 50

**1 mark** Screenshot shows the encoded message correctly



## Task 10 (max. 13 marks)

**1 mark** Print statement appears after the one to print out the message in Morse code

**1 mark** Message prints out the value from the call to CalculateTransmissionTime

**1 mark** Variable MorseCodeString correctly passed as the argument

****

**1 mark** Subroutine takes one parameter which has a meaningful identifier

**1 mark** There is a variable to hold the total transmission time which is initialised to 0

**1 mark** There is an iterative statement to loop through the entire message

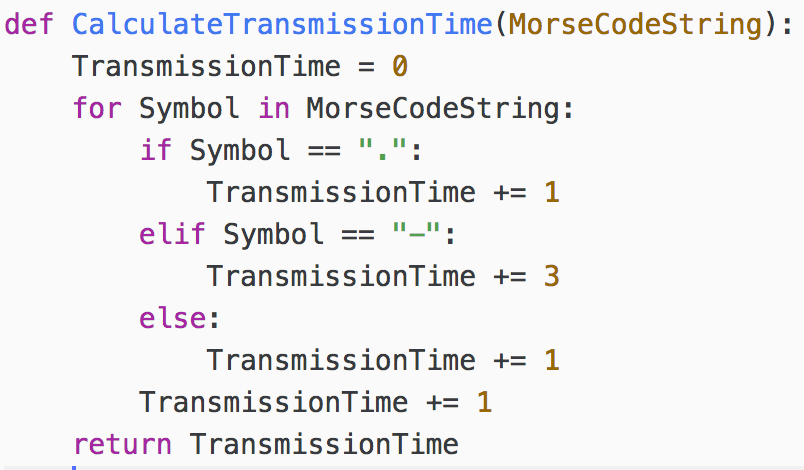
**1 mark** There is a selection statement inside the iterative statement

**1 mark** The selection statement adds 1 for a dot and 3 for a dash

**1 mark** There is an additional +1 time unit after every dot or dash

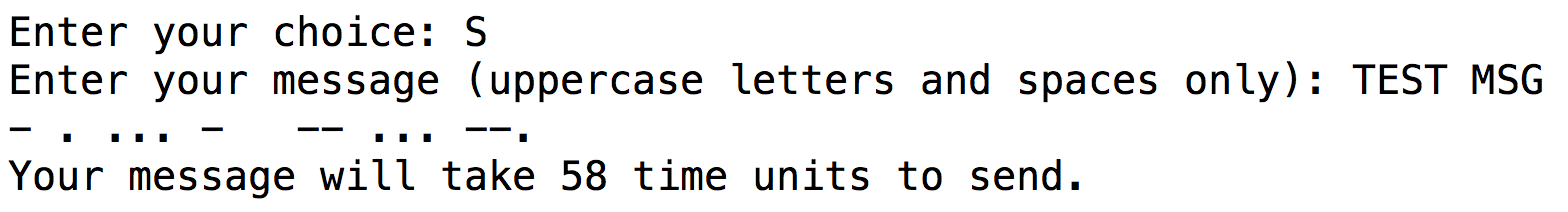
**1 mark** The total additional time for an end of letter is +3

**1 mark** The total additional time for an end of word is +7



**1 mark** Screenshot show S being chosen from the menu and the message TEST MSG being entered

**1 mark** Screenshot shows 58 time units (after the Morse code)

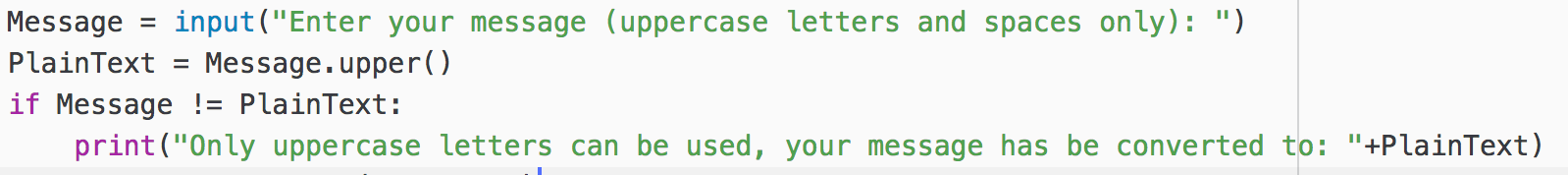


## Task 11 (max. 6 marks)

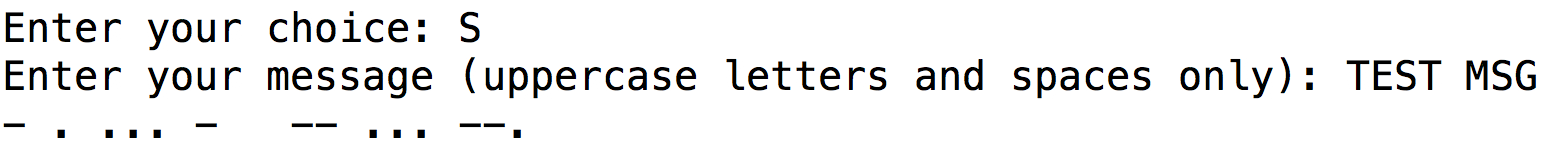
**1 mark** Message is not converted to uppercase as it is input

**1 mark** Selection statement comparing the message to an uppercase version of the message (or checking if the message contains at least one lowercase letter)

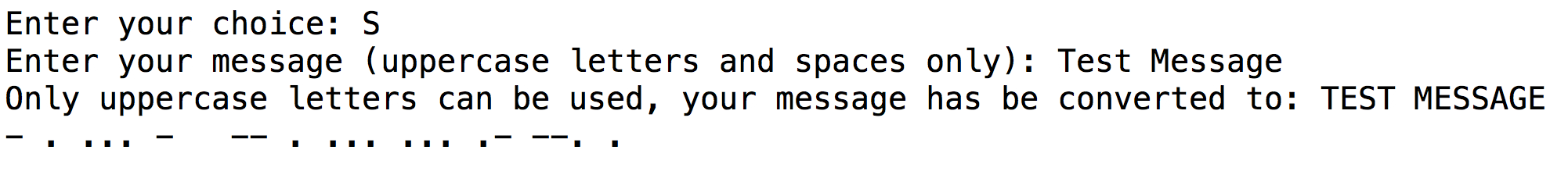
**1 mark** Selection statement contains a print statement which explains that the message has been converted and shows the uppercase message



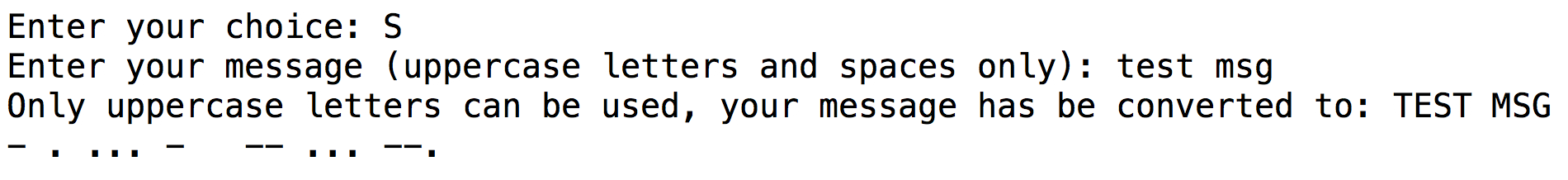
**1 mark** Screenshot shows as below (with no message about converting it)



**1 mark** Screenshot the message converted to uppercase including an explanatory message



**1 mark** Screenshot the message converted to uppercase including an explanatory message



## Task 12 (max. 6 marks)

**1 mark** Print statement appears after the iterative structure that parses the message

**1 mark** Number of symbols computed either by counting the number of dots and dashes or by taking the length of the MorseCodeString and deducting the number of spaces (or by some other reasonable means)

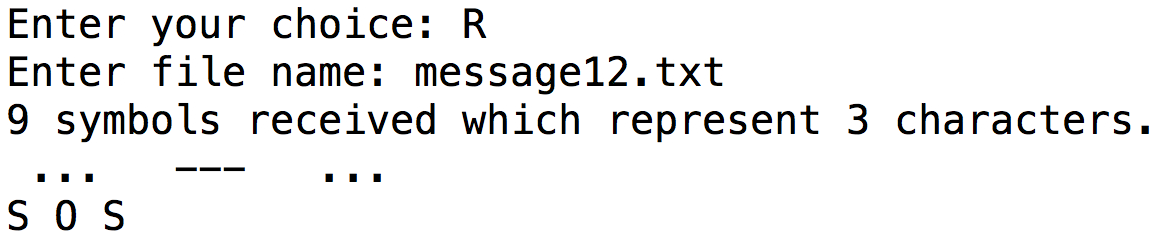
**1 mark** Number of characters computed either by counting the number of letters from A-Z or by taking the length of PlainText and deducting the number of spaces (or by some other reasonable means)

**1 mark** Print statement is of exactly the same format as the question with the correct spacing and capitalisation



**1 mark** Screenshot shows five lines of messages of similar content and format to those shown below in the SAME ORDER as those shown below

**1 mark** Screenshot shows 9 symbols received and 3 characters received



## Task 13 (max. 17 marks)

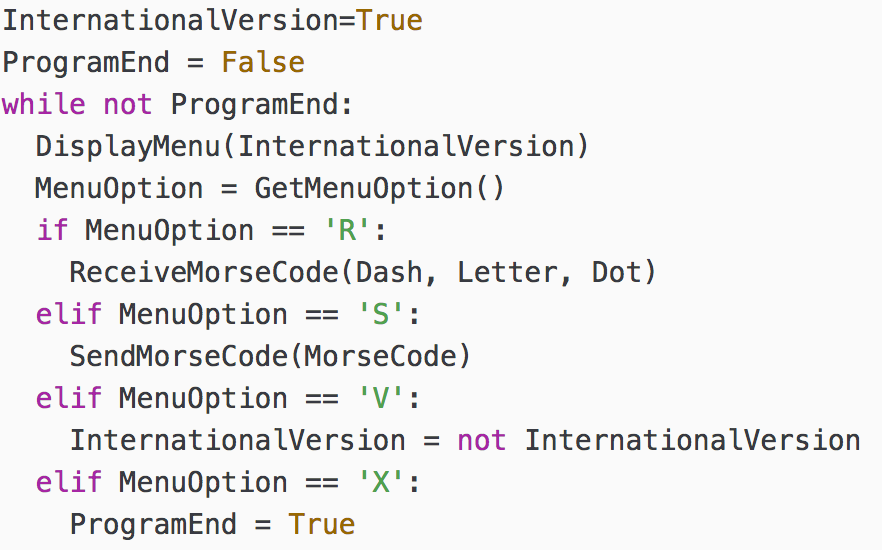
**1 mark** New variable created with a sensible identifier for InternationalVersion

**1 mark** Variable is defined and initialised to True within SendReceiveMessages

**1 mark** Call to DisplayMenu now passes the argument InternationalVersion

**1 mark** Menu option V is added to the selection statement

**1 mark** Selection statement for option V changes the value of InternationalVersion from True to False or vice-versa



**1 mark** New parameter for DisplayMenu added with meaningful identifier

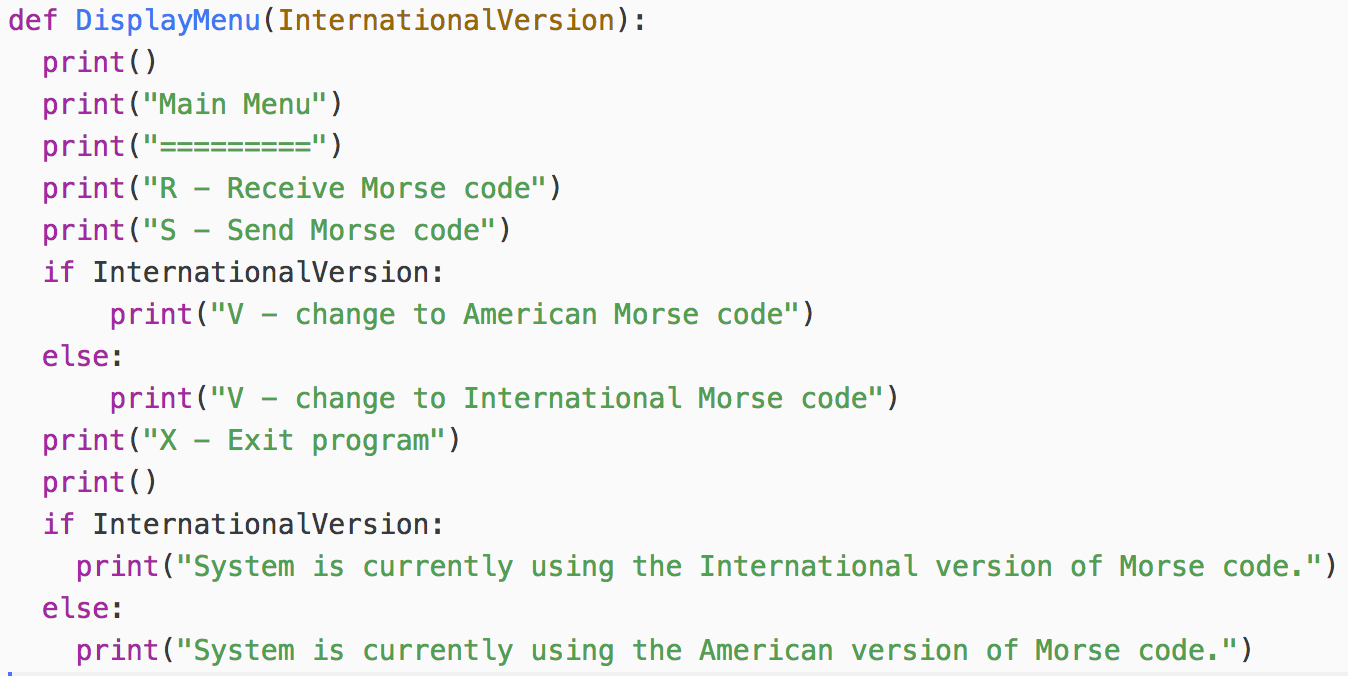
**1 mark** Selection statement added for InternationalVersion

**1 mark** Selection statement affects what is displayed on the menu

**1 mark** Menu options refer to either American version or International version

**1 mark** After the menu has printed, there is another selection statement for InternationalVersion

**1 mark** Selection statement will print out a suitable message according to the value of InternationalVersion correctly stating which version of Morse code is being used



**1 mark** Screenshot shows menu option V has been added

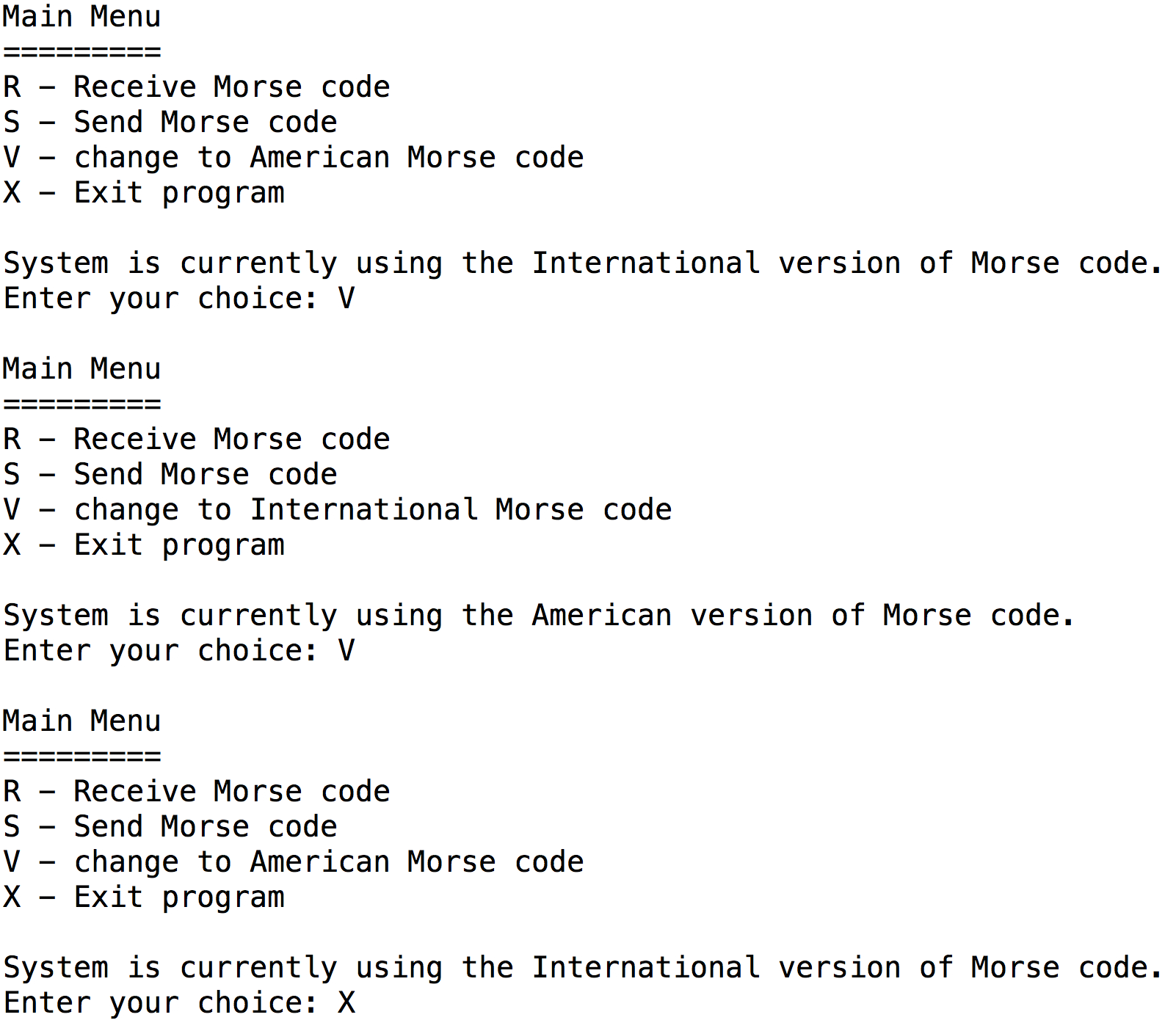
**1 mark** First menu refers to change to American Morse code

**1 mark** Screenshot shows that the initial version of Morse code is the International one

**1 mark** Screenshot shows that V was selected from the first menu

**1 mark** Screenshot shows that the menu option correctly toggles to International version for the second menu

**1 mark** Screenshot shows that the message correctly toggles from International version after the first menu to American version after the second and then back again after the third



## Task 14 (max. 11 marks)

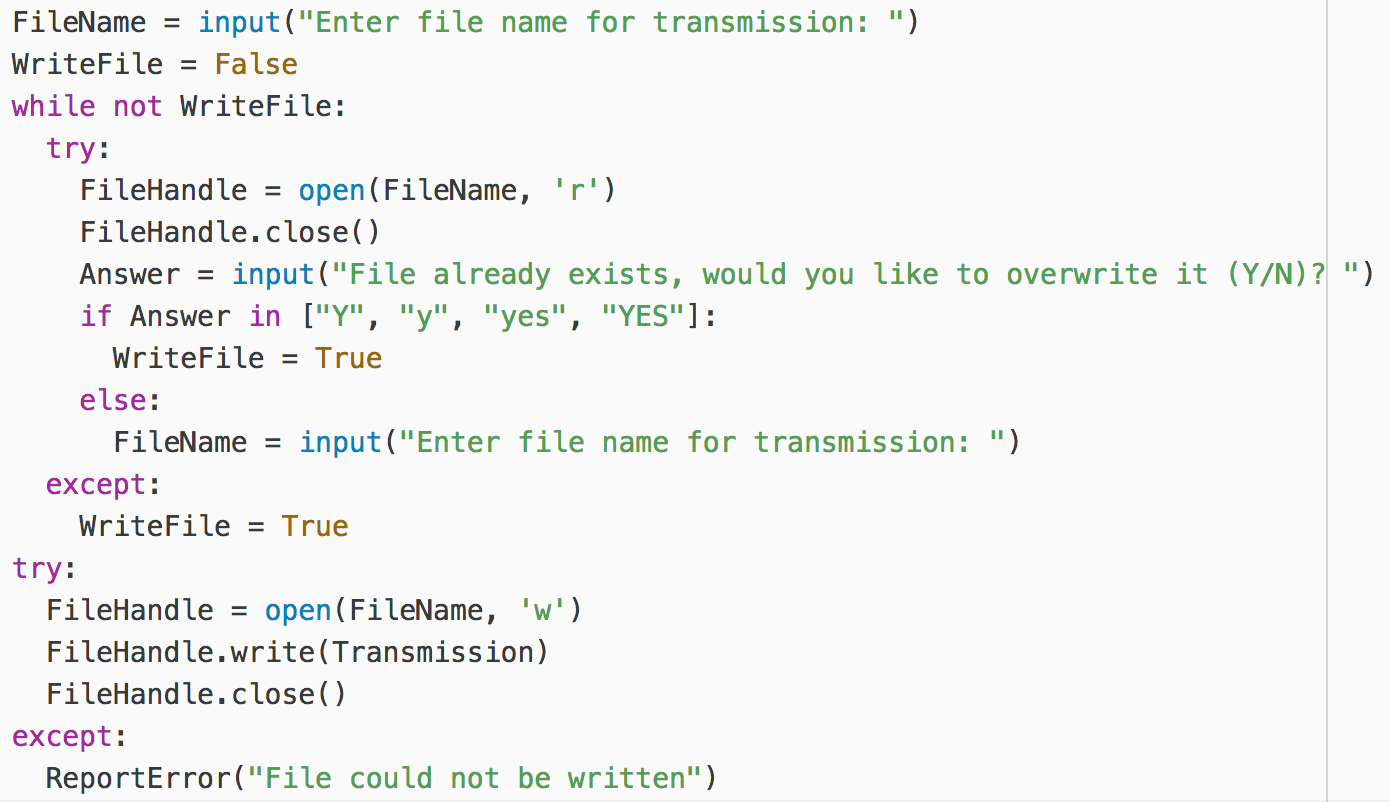
**1 mark** Iterative statement with a sensible condition to keeping checking until a suitable file name is entered or the user chooses to overwrite the file

**1 mark** Structure such as try… except… with an open statement which tests if the file exists or not

**1 mark** Prompt asking the user if they would like to overwrite the file or not and advising them that it already exists

**1 mark** Selection statement exits the loop if they want to overwrite the file

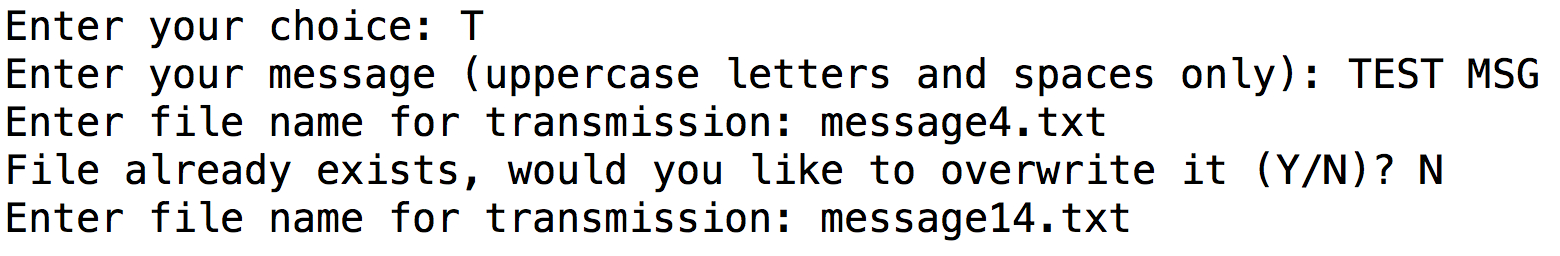
**1 mark** Selection statement asks for a new file name if they don’t want to overwrite the file



**1 mark** Screenshot shows user entering T and then TEST MSG correctly

**1 mark** User enters message4.txt and the program responds with file already exists message asking if you would like to overwrite it

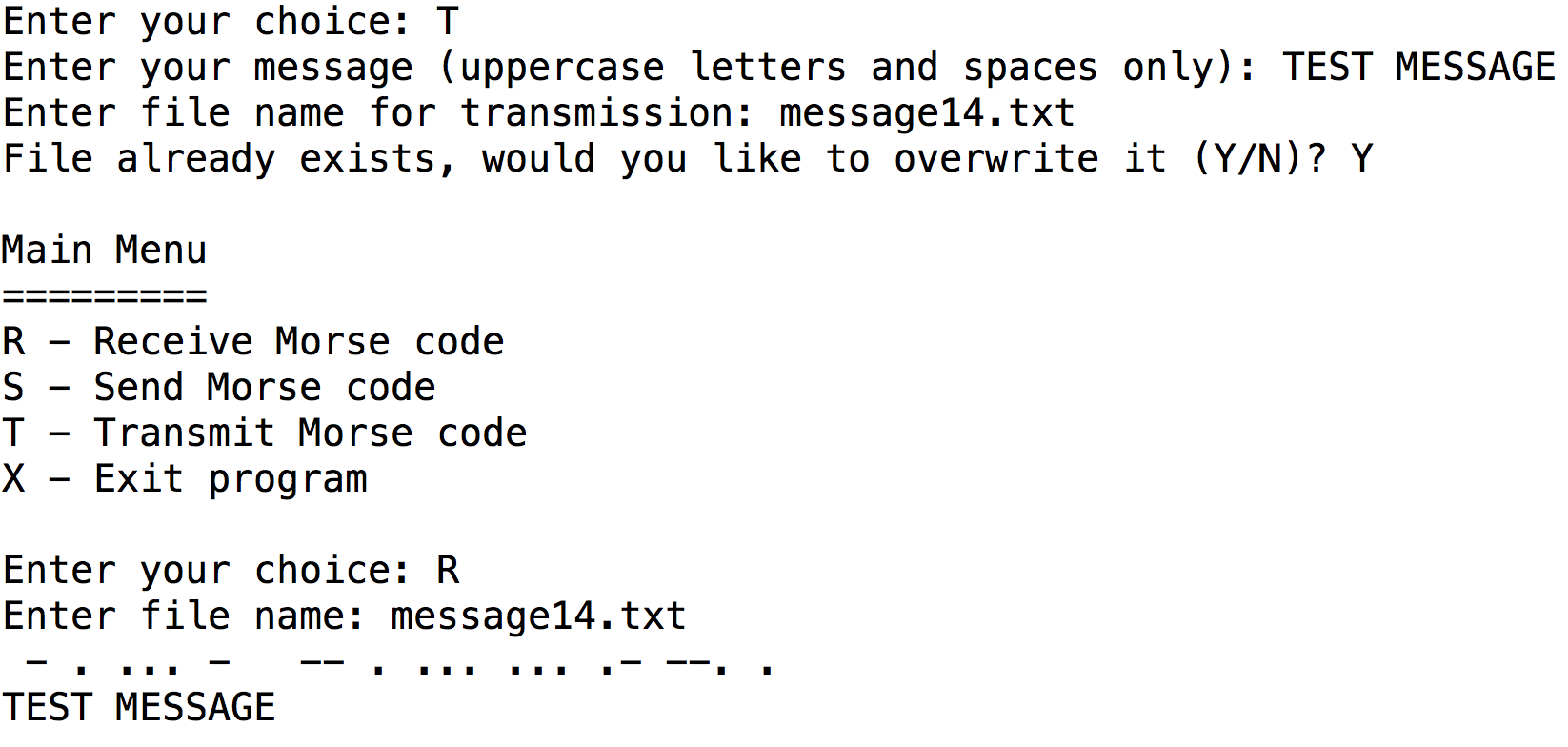
**1 mark** User selects N and enters message14.txt which results in the output shown below (user is taken back to the main menu)



**1 mark** Screenshot shows user entering T and then TEST MESSAGE correctly

**1 mark** User enters message14.txt and then Y – program responds with the main menu

**1 mark** User selects R from the main menu and message14.txt which results in the output shown below



## Task 15 (max. 7 marks)

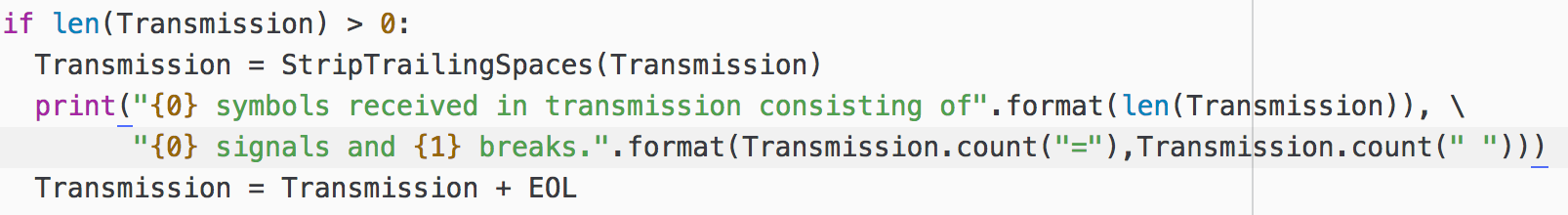
**1 mark** Print statement is inside the selection statement shown below

**1 mark** Message prints out the length of Transmission as the total number of symbols received

**1 mark** Message correctly counts the number of “=” in the Transmission

**1 mark** Message correctly counts the number of “ ” in the Transmission

**1 mark** Message printed is of the correct format and matches the example in the question exactly



**1 mark** Screenshot shows 33 symbols received in total

**1 mark** Screenshot shows that there were 16 signals and 17 breaks

