AQA

AS Computer Science

Paper 1

June 2018

Preliminary Material

To be opened and issued to candidates on or after 1 March 2018 subject to the instructions given in the Teachers' Notes (7516/1/TN).

Note

- The **Preliminary Material**, **Skeleton Program** and **Data File** are to be seen by candidates and their teachers **only**, for use during preparation for the examination on **Monday 4 June 2018**. It **cannot** be used by anyone else for any other purpose, other than that stated in the instructions issued, until after the examination date has passed. It must **not** be provided to third parties.

Information

- A Skeleton Program is provided separately by your teacher and must be read in conjunction with this Preliminary Material.
- You are advised to familiarise yourselves with the Preliminary Material and Skeleton Program before the examination.
- A copy of this Preliminary Material and the Skeleton Program will be made available to you in hard copy and electronically at the start of the examination.
- You must **not** take any copy of the Preliminary, Skeleton Program and Data File or any other material into the examination room.

INSTRUCTIONS FOR CANDIDATES

The question paper is divided into **three** sections.

Section A

You will be asked to create a new program and answer questions **not** related to the **Preliminary Material** or **Skeleton Program**.

Section B

Questions will refer to the **Preliminary Material** and the **Skeleton Program**, but will not require programming.

Section C

Questions will use the **Preliminary Material** and the **Skeleton Program** and may require the **Message.txt Data File**.

Electronic Answer Document

Answers for **all** questions, for **all** sections, must be entered into the word-processed document made available to you at the start of the examination and referred to in the question paper rubrics as the **Electronic Answer Document**.

Preparation for the Examination

You should ensure that you are familiar with this **Preliminary Material** and the **Skeleton Program** for your programming language.

Morse code

The **Skeleton Program** accompanying this **Preliminary Material** is a program for converting between plain text and Morse code.

Samuel Morse invented a code that allowed the transmission of text using simple on-off signals. The code differentiates between short and long signals, usually represented as dots and dashes.

For example, ships within line of sight could use a light turned on and off to communicate with each other. Early telegraph communication used electrical pulses that were converted into sound.

Each letter of the alphabet is represented by a unique combination of dots and dashes. Different letters may have a different number of dots and/or dashes. For example, the letter T is represented by a single dash and the letter H is represented by four dots.

Morse codes for each letter in the alphabet are shown in **Table 1**.

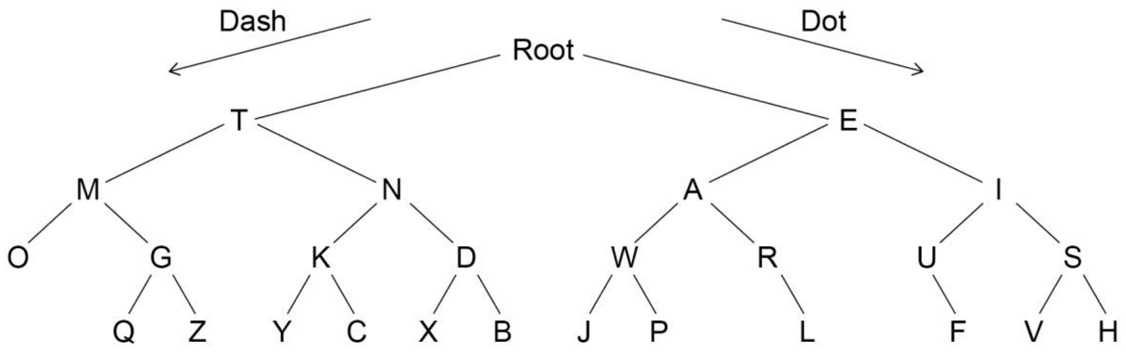
Table 1

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

The codes can be represented using a tree, as shown in **Figure 1**. Starting at the root of the tree, taking a left branch generates a dash and taking a right branch generates a dot.

For example, the letter D has a Morse code generated by first taking the left branch (dash), then two right branches (dot, dot). So the code for the letter D is dash, dot, dot.

Figure 1



However, there can be confusion over where letters start and end. For example, the signals for the letters TEA (T is dash; E is dot and A is dot, dash) are the same as for the letter X (X is dash, dot, dot, dash).

▬●●▬

To indicate where a letter ends when writing Morse codes, a space is placed between signals representing letters, shown in this Preliminary Material with a ⌂ character. So TEA becomes:

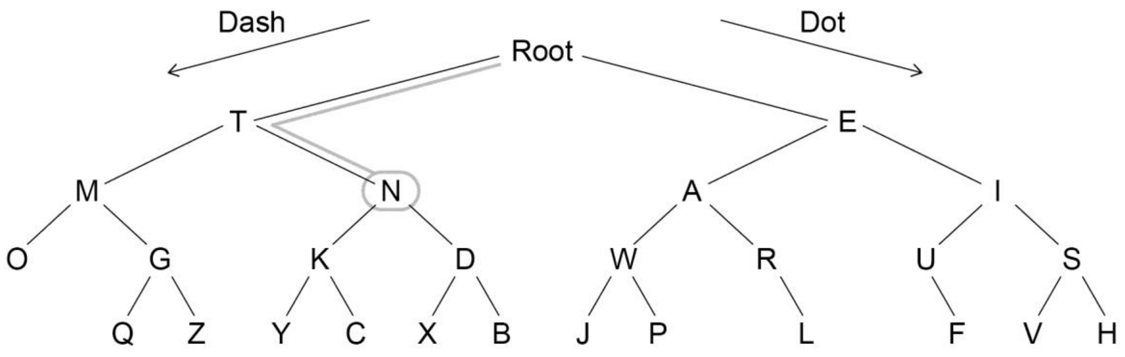
▬⌂•⌂•▬

A Morse code message can be decoded using the diagram in **Figure 1**. For each new group of dots and/or dashes, start at the root of the tree. When reading a dash, take the left branch, when reading a dot, take the right branch. When the end of the signal is reached the node of the tree reached represents the letter corresponding to that signal.

For example, the message ▬●⌂▬▬▬ is decoded as shown in **Figure 2** and **Figure 3**.

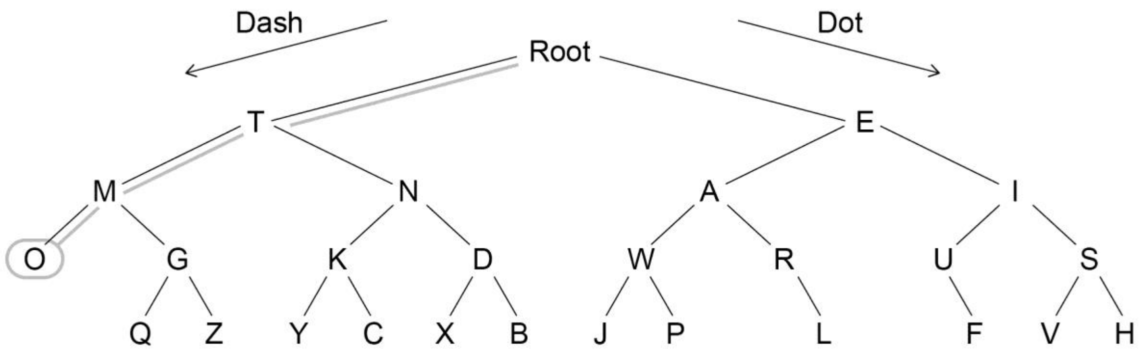
The group ▬● decodes as N:

Figure 2



The group ▬▬▬ decodes as O:

Figure 3



When writing Morse code, each letter is separated from the next letter by one space and a word is separated from the next word by three spaces. There are no spaces after the last letter in a message.

Two English messages and their Morse codes are shown in **Table 2.**

Table 2

English message	Morse code
SOS	...⌂---⌂...
TEA X	-⌂.⌂.-⌂⌂⌂-..-

When using light or sound to transmit Morse code, a short pause is required between two signals. The rules of transmission when using light or sound are:

- a transmission may have leading and trailing spaces
- a transmission consists of time units, each of which is either on or off (light/sound or no light/sound)
- a dot is one time unit of light/sound followed by an interval of no light/sound lasting one time unit
- a dash is three time units of light/sound followed by an interval of no light/sound lasting one time unit
- There is an interval of no light/sound between characters lasting an extra two time units (making 3 time units in total between characters)
- There is an interval of no light/sound between words lasting an extra four time units (making 7 time units in total between words – 1 unit after the dot/dash, 2 extra units after the character and 4 extra units after the word)

Examples of transmission signals are shown in **Table 3**.

Table 3

Message	Morse code	Transmission signals
SOS	...⌂---⌂...	=⌂=⌂=⌂⌂⌂===⌂===⌂===⌂⌂⌂=⌂=⌂=⌂
TEA X	-⌂.⌂.-⌂⌂⌂-..-	===⌂⌂⌂=⌂⌂⌂=⌂===⌂⌂⌂⌂⌂⌂⌂===⌂=⌂=⌂===⌂

The transmission signal contained in the **Data File Message.txt** is shown in **Figure 4**.

Figure 4

Index	Signal
0	=
1	=
2	=
3	⌂
4	⌂
5	⌂
6	=
7	⌂
8	⌂
9	⌂
10	=
11	⌂
12	=
13	=
14	=
15	⌂
16	⌂
17	⌂
18	⌂
19	⌂
20	⌂
21	⌂
22	=
23	=
24	=
25	⌂
26	=
27	⌂
28	=
29	⌂
30	=
31	=
32	=
33	⌂

Turn over ►

The **Skeleton Program** stores each letter of the alphabet in the data structure `Letter`. The Morse code for each letter is stored in the data structure `MorseCode`.

Index	**Letter**	**MorseCode**
0	SPACE	SPACE
1	A	.—
2	B	—...
3	C	—.—.
4	D	—..
5	E	.
6	F	..—.
7	G	——.
8	H	....
9	I	..
10	J	.———
11	K	—.—
12	L	.—..
13	M	——
14	N	—.
15	O	———
16	P	.——.
17	Q	——.—
18	R	.—.
19	S	...
20	T	—
21	U	..—
22	V	...—
23	W	.——
24	X	—..—
25	Y	—.——
26	Z	——..

The data structure `Letter` stores the data shown at the nodes of the tree in **Figure 1**. Two further data structures, `Dash` and `Dot`, are used to represent the information on how the nodes are connected. These connections are known as pointers.

For example, the letter A is stored at index 1 in `Letter`. The left branch from A goes to W, stored at index 23, so `Dash` at index 1 stores the pointer value 23. The right branch from A goes to R, stored at index 18, so `Dot` at index 1 stores the pointer value 18.

When there is no node to the left or right, the pointer value is 0 (zero).

Index	Dash	Letter	Dot
0	20	SPACE	5
1	23	A	18
2	0	B	0
3	0	C	0
4	24	D	2
5	1	E	9
6	0	F	0
7	17	G	26
8	0	H	0
9	21	I	19
10	0	J	0
11	25	K	3
12	0	L	0
13	15	M	7
14	11	N	4
15	0	O	0
16	0	P	0
17	0	Q	0
18	0	R	12
19	22	S	8
20	13	T	14
21	0	U	6
22	0	V	0
23	10	W	16
24	0	X	0
25	0	Y	0
26	0	Z	0

Turn over ►

The **Skeleton Program** presents the user with a menu that allows the user to choose between sending and receiving a message using Morse code.

Receiving a message involves reading a text file containing a transmission string (see **Figure 4**) and converting this first into dots and dashes and then into letters of the alphabet.

Sending a message asks the user to enter a message using uppercase letters and spaces. The program will then output the dots and dashes of the equivalent Morse code message, including the correct number of spaces between characters and words.

END OF SOURCES