RECOGNISING ACHIEVEMENT

Candidate Style Answers

OCR GCSE ICT J461 / J061

Unit B065 Practical Applications in ICT: Controlled Assessment Task

High Level Candidate Response

This support material booklet is designed to accompany the OCR GCSE ICT specifications for teaching from September 2010.

Introduction

OCR has produced these candidate style answers to support teachers in interpreting the assessment criteria for the new GCSE specifications and to bridge the gap between new specification release and availability of exemplar candidate work.

This content has been produced by subject experts, with the input of Chairs of Examiners, to illustrate how the sample assessment questions might be answered and provide some commentary on what factors contribute to an overall grading. The candidate style answers are not written in a way that is intended to replicate student work but to demonstrate what a "good" or "excellent" response might include, supported by examiner commentary and conclusions.

As these responses have not been through full moderation and do not replicate student work, they have not been graded and are instead, banded "middle" or "high" to give an indication of the level of each response.

Please note that this resource is provided for advice and guidance only and does not in any way constitute an indication of grade boundaries or endorsed answers.

Task

The task:

A teacher at a local primary school wants to help some students improve their spelling. These students will take time away from the class and work by themselves with a computer program that will help them to spell simple words from the class spelling lists for that week. The teacher wants to be able to specify the words for each student by entering them or by selecting them individually from a list.

The interface must be colourful and easy to use for the student; it needs to report back to the student on their success. The teacher also needs to know how well the student has done after each test by identifying words that caused problems and a score. The teacher interface needs to be simple to use.

Your task is to create a suitable solution with the basic functionality as a priority.

Analysis:

What do I know?

- My end user is a teacher in a primary school
- The system is to help 'some' students with their spelling
- The spellings are for simple words
- There is a class list for the week
- The teacher wants to select the words
- The interface should be colourful and easy to use
- The system should tell the student if they are right or not
- The teacher wants to see how well the student has done
- The teacher wants to know which words the student found difficult
- The system needs a simple teacher interface
- Basic functionality is the priority

What don't I know?

- The age of the students
- Can the students use a keyboard?
- What sort of words are in the lists
- The style of interface required
- How the system reports back to the student
- How the system reports back to the teacher
- How the teacher wants to input the words
- How many words / wordlists does the teacher require
- How the student can be identified by the system

Questions to ask the end user:

1. What age are the students in this group?
2. The system is for individual use. What is the reason for this?
3. What words are in the lists?
4. How do you want the system to ask for the spelling?
5. How do you want to input the words to be used?
6. How many word lists do you require
7. How do you want to keep track of student progress?
8. What hardware is available in the classroom?
9. Do you want sound?
10. What software is available

(See attached notes for discussion and responses.)

Analysis of responses:

This is a system for individual use with the student sitting at a computer copying words and learning how these sound. They student will be using a large key touch pad to enter spellings so will not have access to a mouse or any control characters or function keys. This means I will need to use a menu interface.

The teacher wants sets of standard wordlists and will not want to repeatedly type these into the system. Hence these need to be stored on the PC under meaningful file names entered by the teacher. There may be one test per week so up to 40 tests for a school year

I plan to write a program so apart from anything required within that environment there is no need for any specific software on the system. A free programming environment will be useful and one I can load onto the target PC as necessary. The XP operating system poses few difficulties.

The teacher suggested a system where the word is displayed, spoken then copied by the student. The system should then respond to the student and should display a score. The teacher would like to have a record of which words the student found difficult.

What I need to do now:

- I need to identify existing systems for suitable designs and approaches
- I need to identify a way to get the computer to say the words
- I need to create some designs and get these ideas back to the teacher

1. Age? (T) just learning to read and write
(T) simple sounds to recognise.

2. Individual use – why?
(T) – Extra practice to recognise words and match these to sounds + spellings.

3. Single words? (T) – yes to teach vowel sounds such as a A (short and long A's) 'ee', e so :-
cat bat ball feet etc.

4. How? (T) – they are learning at the moment so they really need to see, copy and hear
(M) – so! print word on screen, speak the word + they can type it in?
(T) – yes
(M) – can they use keyboards?
(T) – we have some large keypad devices they can use so we can only input letters and numbers
(M) – Do these work with the standard PC?
(T) – yes – just no control characters etc.

5. Words? (T) – I will type the words in for the weekly tests – they will be themed eg short a/ long a etc.

6. How Many lists? (T) Not sure really – we have lots so as many as we can
(M) – If I set up a file system can you create separate files for the word sets?.

7 - Keeping track?

(T) - feedback to the student is most important, I don't really need a score but it would be useful to see which words cause problems.

(M) - If I stored the question + answer would that be useful?

(T) - Yes - but not vital - I will probably keep an eye on them anyway and they will do the weekly tests for more detailed feedback.

8 - Hardware? (T) - Standard PC with XP, printer, large touch sensitive keypad, speakers, headphones.

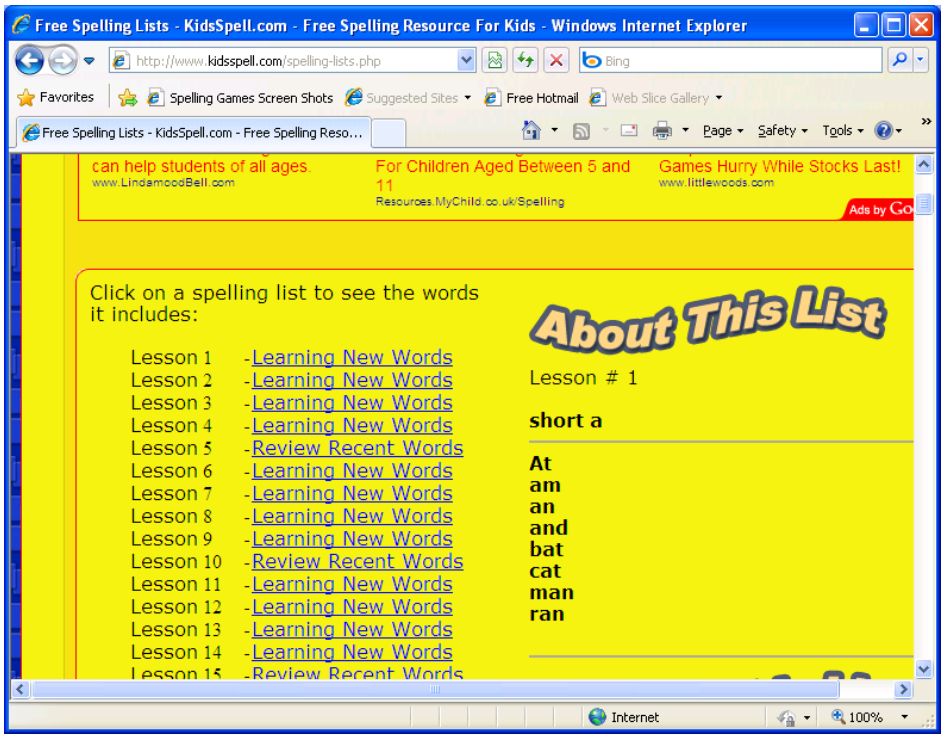
9 - Sound? (T) - Yes they need to match the sound with the spelling - Headphones best!

10 - Software? (T) Usual stuff - office etc, some Teaching + learning applications. Does this make a difference?

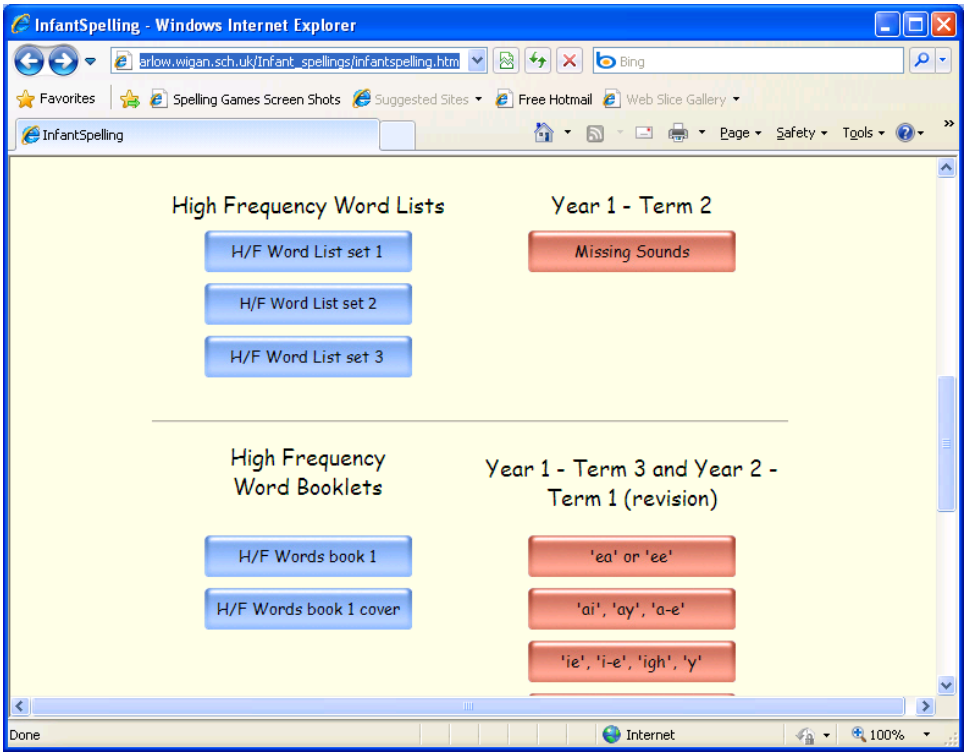
(M) Not significant - I would like to write a program for this. Is that ok?

(T) Yes, sound good - get back to me with your ideas.

http://www.kidsspell.com/spelling-lists.php



http://www.saintambrosebarlow.wigan.sch.uk/Infant_spellings/infantspelling.htm



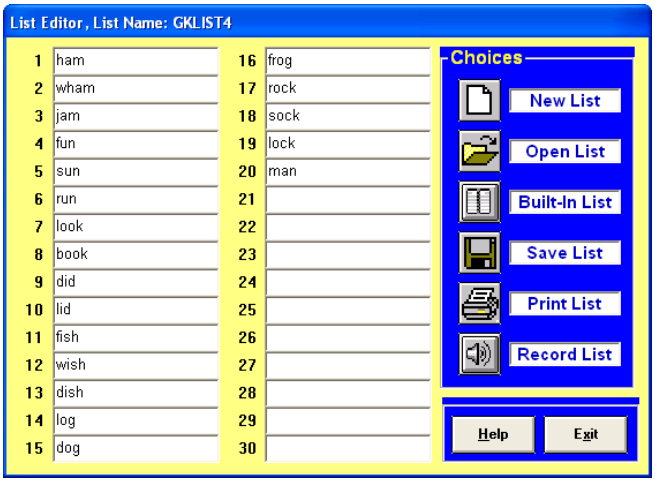
These sites indicate the sort of words I might need but do not provide any guidance on the type of interface that would be suitable.

http://www.bbc.co.uk/schools/ks1bitesize/literacy/spelling/index.shtml

The BBC site uses a fill in the gap approach to teaching simple word sounds.

This is similar to what the teacher wants:

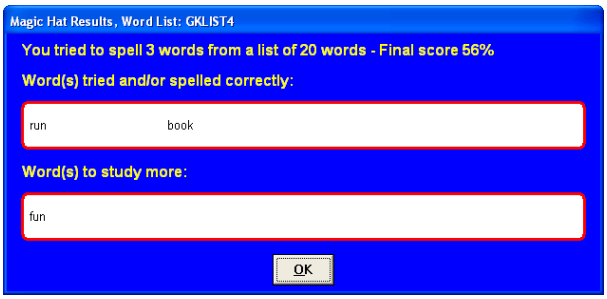
The spelling games from kidwaresoftware include other potential systems with word lists



AND displays the word for a short while so the student can copy it.

The teacher did not imply that they wanted such a system so I think simply keeping it displayed will be more appropriate.

These ‘games’ also provide feedback on words that caused problems.

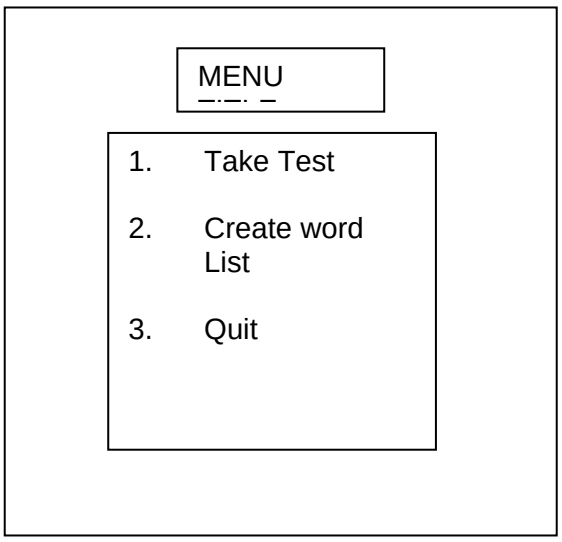


I now have some ideas about how I can design my system for the teacher.

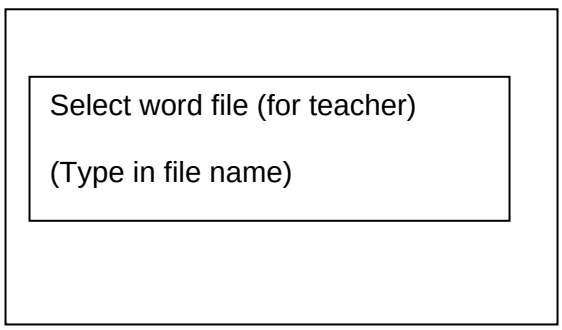
The design requirements for my system are:

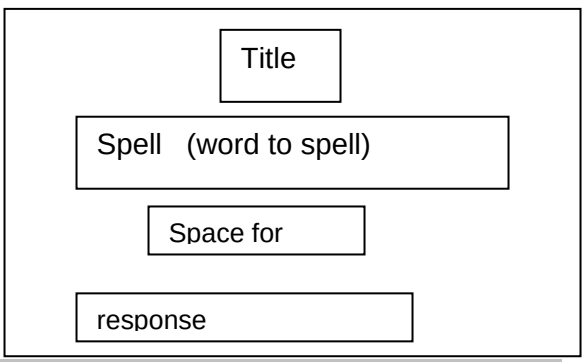
- Simple menus.
- Teacher routine to create wordlists and save as files to be loaded
- Words to be displayed and sounded
- Simple clear interface
- Student to type word and get feedback
- Overall score for student to be displayed
- Words that caused problems to be identified for the teacher

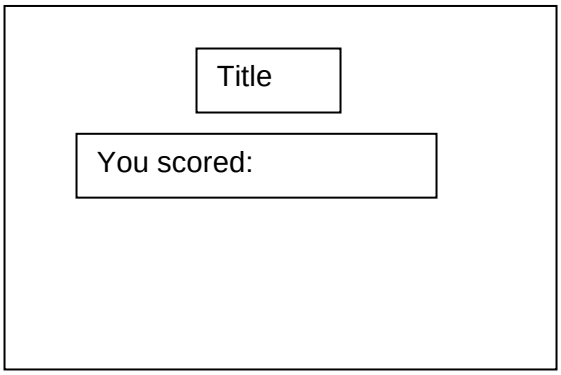
Screen designs: 1.Menu



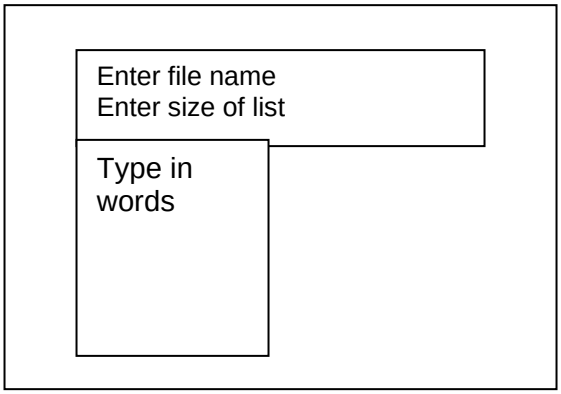
2. Take Test







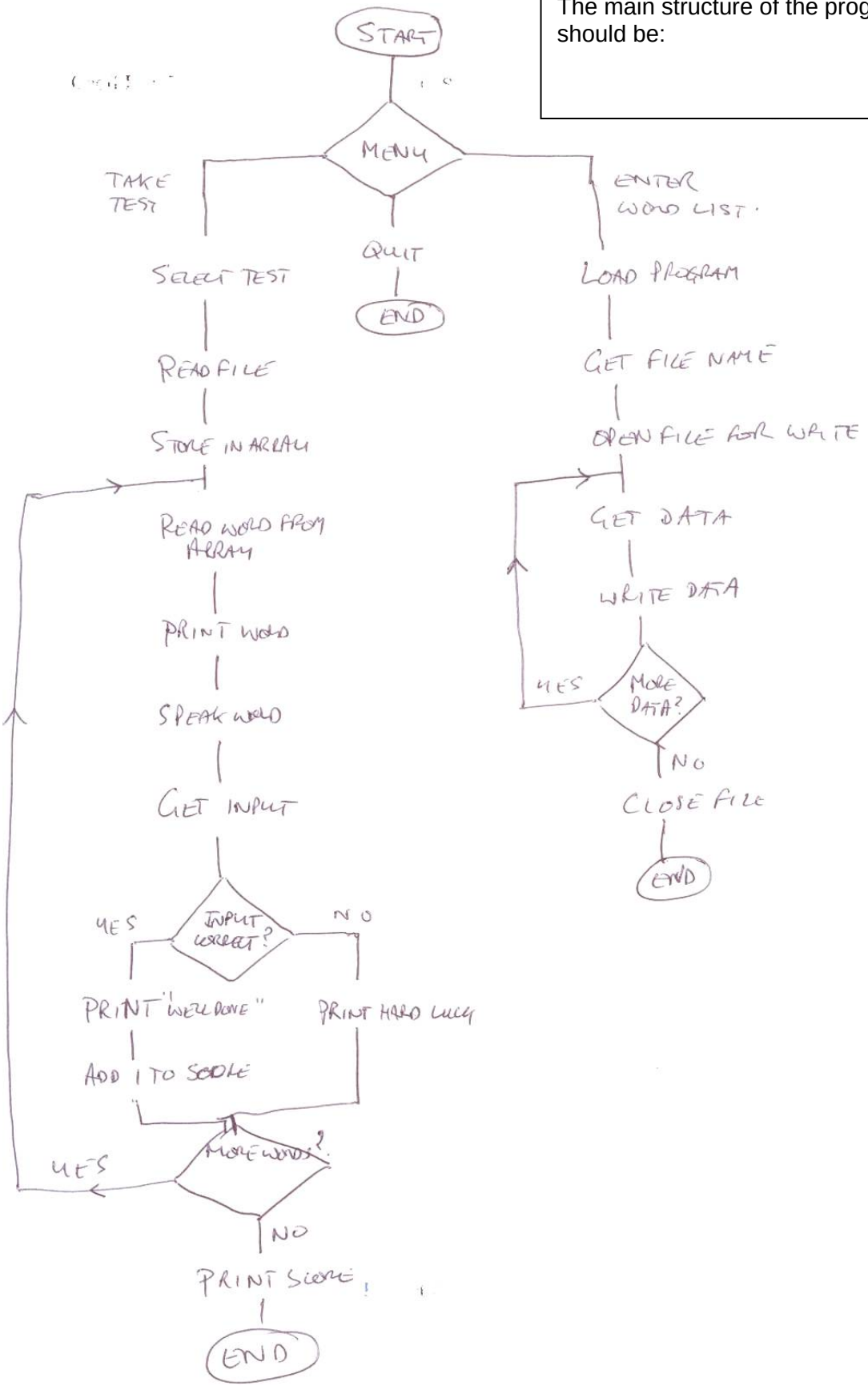
3 Create word list

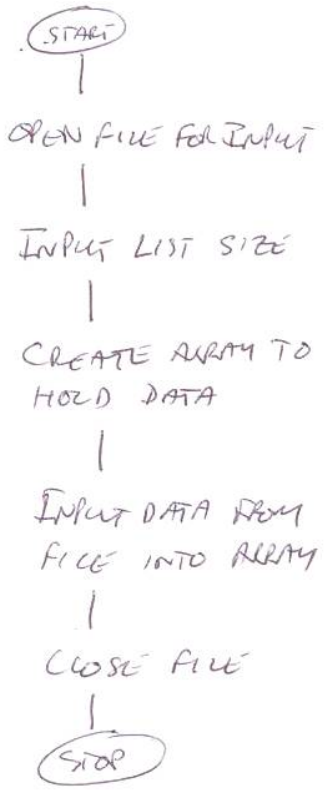


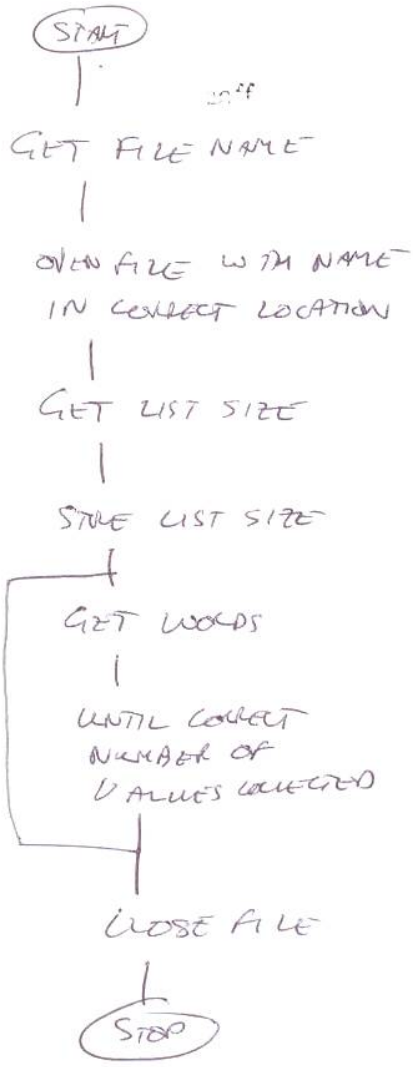
The screen colours need to contrast with the text colour and it needs to be clear where to type in the word for the student.

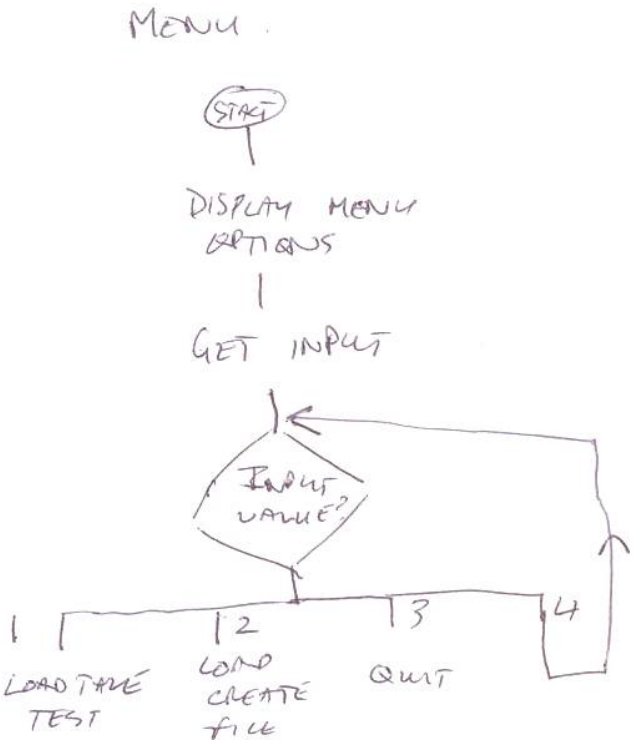
Some on screen instructions are needed but since the student is only just learning to read there is little value in these on the student test page. The teacher will not need many instructions since the process needs to be straightforward and there are few things for them to do.

The main structure of the program should be:









To include storing incorrect responses.

- Add option get student name or id
- Create an array the same size as the word list
- Initialise a counter
- If word incorrect write to array and increment counter
- When test complete
- Open a file using student name for write
- Write contents of array to file
- Close

In menu program add another option to view last test

- Load a program to open a file for input
- Read in data until EOF
- Display
- Close file

Actions:

- Identify how to make computer say words
- Identify how to create files from input string
- Identify how to read and write data from and to files
- Create characters to indicate correct or incorrect

Test strategy:

I will use test data to check that each element of the code works as expected as I develop it using valid data until the routine works then testing with extreme and invalid data. I will create a data file called words1 with four words based on the short a, ie cat, bat, hat and sat.

Once a routine works I will add the next component and repeat the process until all components are included.

I will test the final product against the design criteria and for basic functionality.

Test	Data	Reason	Expected	Actual
Menu	1,2,3 and 4	Valid options	Does as expected	
	5,-1,a	Invalid data	Error message and return to menu	
Create word list	Words1, 4, cat, hat, sat, mat	Valid data	Should create a file called words1 with 4 data items	
Take Test	Danny, words1	Valid data	Should load the wordlist words1, complete test and display score, and create a file called danny with incorrect responses	
	Danny, words1, cot, hot sat mat	Valid data, two correct responses	Should score 2 out of 4 and file danny should contain the words cat and hat	
	Danny, noexist	Invalid file name	Should display error message about a no existent word list	

This means I need to investigate how to code spoken prompts for the words.

I have some experience with BBC Basic for Windows so I looked at various sites for ideas.
www.bbcbasic.co.uk and www.bb4w.com which led to various links
I could dismiss many of them immediately:
For sale on this site is a **CD Book** containing over 100 *BBC BASIC for Windows* programs with answers and explanations. Also available are resources for digital logic design and simulation.

Jon Ripley

An introduction to writing *BBC BASIC* applications which make use of the Windows Application Program Interface, including a reference guide to data types, links to other useful documents, example code and libraries.

Gordon Sweet

A fairly large collection of old Games, Animations, BBC Music and other useful programs.

I looked at several of these sites and found a speak text routine on R T Russell's site and further examples of this being used in Gordon Sweet's site.

```
INSTALL @lib$+"COMLIBA"

ON ERROR PRINT 'REPORT$ : PROC_comexit : END
ON CLOSE PROC_comexit : QUIT

filetospeak$ = "C:\test.txt"

PROC_cominit
voice% = FN_createobject("SAPI.SpVoice")
PROC_callmethod(voice%, "Speak("""+filetospeak$+""", 12)")
PROC_releaseobject(voice%)
PROC_comexit
```

This will be my starting point for developing my program.

(There are various standard procedures to include as well as this code)

The teacher asked for a simple and colourful interface.

My research suggests a simple menu will be appropriate

My proposal:

- A programmed solution using BBC Basic.
- A simple menu interface with 4 choices:
    - Take test
    - Create word list
    - View latest test results
    - Quit
- The take test requests student name/id
- The take test requests name of file to use
- The test interface keeps a score
- The system speaks the word using a computer voice
- The word will be displayed for the student to copy
- Simple space to type in the word
- Differing responses for success or failure
- Record incorrect responses for the teacher to view later

Speech procedure:

I have made some minor modifications to the speech procedure I found and have tested it with some normal data.

```
REM. based on program by R.T.Russell, 24-Apr-2007
REM. Error handling routine R.T Russell

INSTALL @lib$+"COMLIBA"

ON ERROR PROC_comexit : PRINT 'REPORT$ : END
ON CLOSE PROC_comexit : QUIT

PROC_cominit

Pitch% = 0
Speed% = 0
Voice$ = ""

REPEAT
  READ word$
  PRINT word$
  PROCspeak(word$,Pitch%,Speed%,Voice$)
UNTIL word$ = ""
PROC_comexit
END

DATA "Cat, Ben, Ball, Feet"
DATA "gtres, bonjour, *&£""
DATA

REM. Procedures from R T Russell

DEF PROCspeak(word$,pitch%,speed%,voice$)
tts% = FN_createobject("Sapi.SpVoice")
IF tts% THEN
  LOCAL qual$
  qual$ = "<PITCH ABSMIDDLE="""+STR$pitch%+"""/><RATE ABSSPEED="""+STR$speed%+"""/>"
  IF voice$<>"" qual$ += "<VOICE REQUIRED="""NAME="+voice$+"""/>"
  PROC_callmethod(tts%, "Speak("""+qual$+word$+""")")
  PROC_releaseobject(tts%)
  ENDPROC
ENDIF
```

```
tts% = FN_createobject("Speech.VoiceText")
IF tts% THEN
  PROC_callmethod(tts%, "Register("""","""COMLIB demo""")")
  PROC_putvalue(tts%, "Enabled(BTRUE)")
  PROC_putvalue(tts%, "Speed("+STR$INT(150*3^(speed%/10))+")")
  PROC_callmethod(tts%, "Speak("""+phrase$+""", 1)")
  REPEAT
    SYS "Sleep", 150
  UNTIL FN_getvalueint(tts%, "IsSpeaking") = 0
  PROC_releaseobject(tts%)
  ENDPROC
ENDIF
```

Abnormal data for this unit would be words which are not standard English words or characters that are not alphabetic.

I used normal data Cat, Ben, Ball, Feet

Abnormal data included gtres, bonjour and *&£”

These abnormal data would probably be the result of typing errors but trapping them would be extremely difficult except for non alphabetic characters, to trap non-words would require the use of a dictionary function.

The normal data caused no problem, but the extra “ in the final abnormal data caused a problem since it was taken as an end “ on the data line leaving the final “ as an unfinished pair of “’s.

Error screen:

Removing that item from the data and using just *&£ the system reads gtres as a word, bonjour works well and the *&£ are read as a string of symbols, ‘asterisk, ‘and’ ‘pound’

I will now modify the routine to use a file instead of the data lines.

```
INSTALL @lib$+"COMLIBA"

ON ERROR PRINT 'REPORT$ : PROC_comexit : END
ON CLOSE PROC_comexit : QUIT

filetospeak$ = "C:\test.txt"

PROC_cominit
voice% = FN_createobject("SAPI.SpVoice")
PROC_callmethod(voice%, "Speak("""+filetospeak$+""", 12)")
PROC_releaseobject(voice%)
```

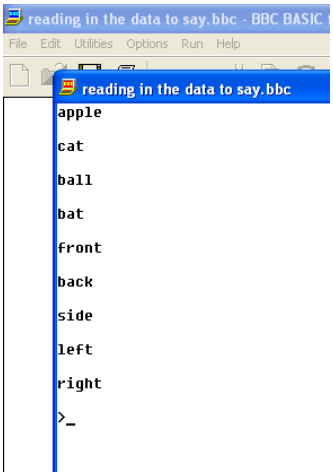
```
PROC_comexit
```

Using standard test data this is able to read a standard text file created using notepad.

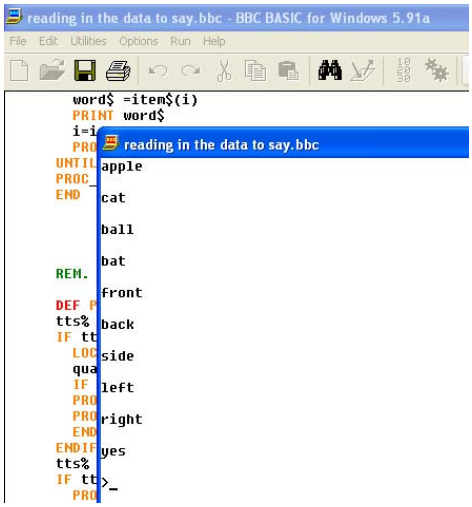
Now I need to get the system to speak individual words from the text file by detecting the breaks between the words. I have taken each item of data from the file and put this into an array to be spoken.

```
DIM item$(10)                          Dimension an array to 10 for the test
a=OPENIN "C:\test.txt"                 Open the test file for input
FOR j=0 TO 10
  INPUT#a,item$(j)                 Loop  to input the ten test values into the array
NEXTj
CLOSE#a                            Close the file
i=0                                Initialise the loop
REPEAT
  word$ =item$(i)                        put each item in turn into a variable and say that variable
  PRINT word$
  i=i+1
  PROCspeak(word$,Pitch%,Speed%,Voice$)        calling the speech routine.
UNTIL word$ = ""
```

The routine speaks all the words except the last one,



I will try adding a blank word at the end of the text file to see if this helps



That worked, the file needs a blank line at the end to allow the program to complete normally.

Now I need to include a routine just after the point where the current program speaks the word to obtain an input from the student and compare it with the correct spelling.

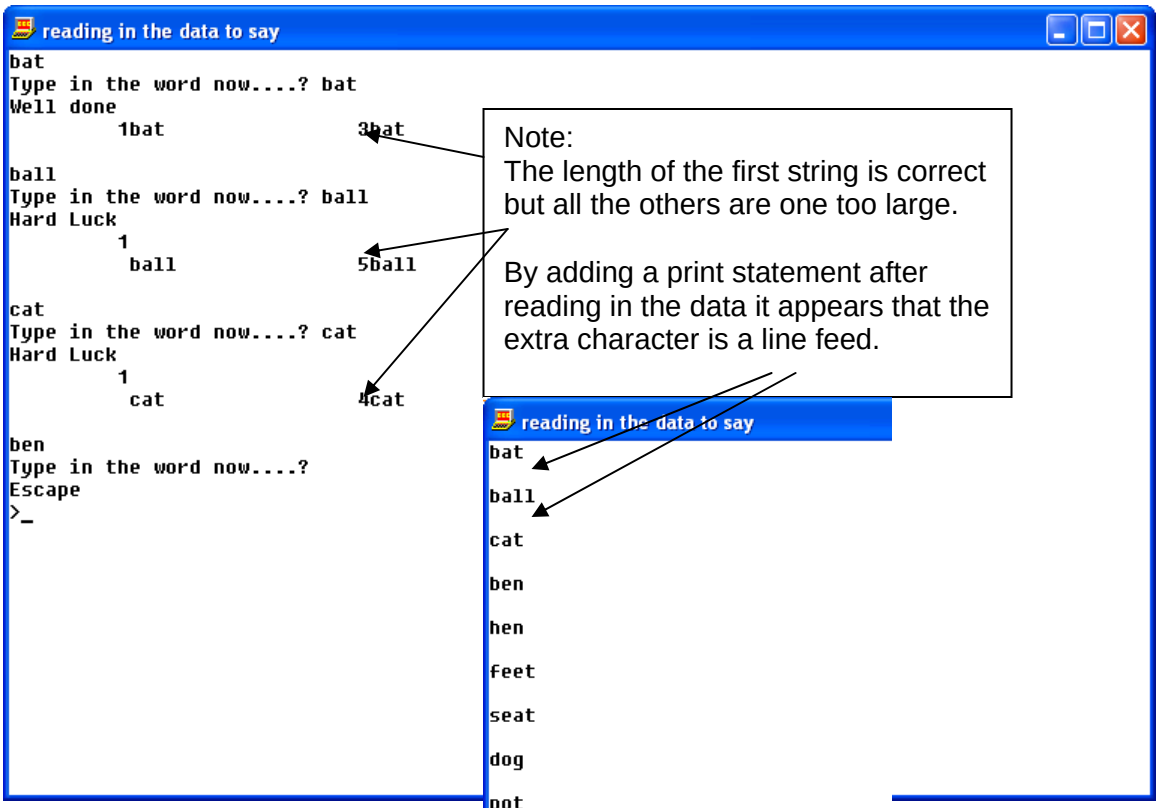
```
DIM item$(10)
correct=0
a=OPENIN "C:\test.txt"
FOR j=0 TO 10
  INPUT#a,item$(j)
NEXTj
CLOSE#a
i=0
REPEAT
  word$ =item$(i)
  PRINT word$
  i=i+1
  PROCspeak(word$,Pitch%,Speed%,Voice$)
  INPUT "Type in the word now....",student$
  IF student$=word$ THEN
    correct=correct+1
    PRINT "Well done"
  ELSE
    PRINT "Hard Luck"
  ENDIF
UNTIL word$ = ""
PRINT "You scored ";correct;" out of ";i
PROC_comexit
END
```

After adding the extra bits of code I tested the routine.
Problems:

- It asks the student to spell the last blank word so I will have to check for this
- The total was not correct

```
PROCspeak(word$,Pitch%,Speed%,Voice$)
INPUT "Type in the word now....",student$(i)
IF student$(i)=item$(i) THEN
  correct=correct+1
  PRINT "Well done"
ELSE
  PRINT "Hard Luck"
ENDIF
PRINT correct, item$(i), LEN(item$(i)),student$(i)
i=i+1
```

By adding this line to print the values of various variables in the program I was able to see that the words in the text file included an extra character.



I created a simple routine to Write a data file to the C drive

```
DIM word$(10)
FOR x= 1TO 10
  INPUT "word ", word$(x)
NEXT
a=OPENOUT "C:/data.txt"
FOR i=1 TO 10
  PRINT#a,word$(i)
NEXT
CLOSE#a
```

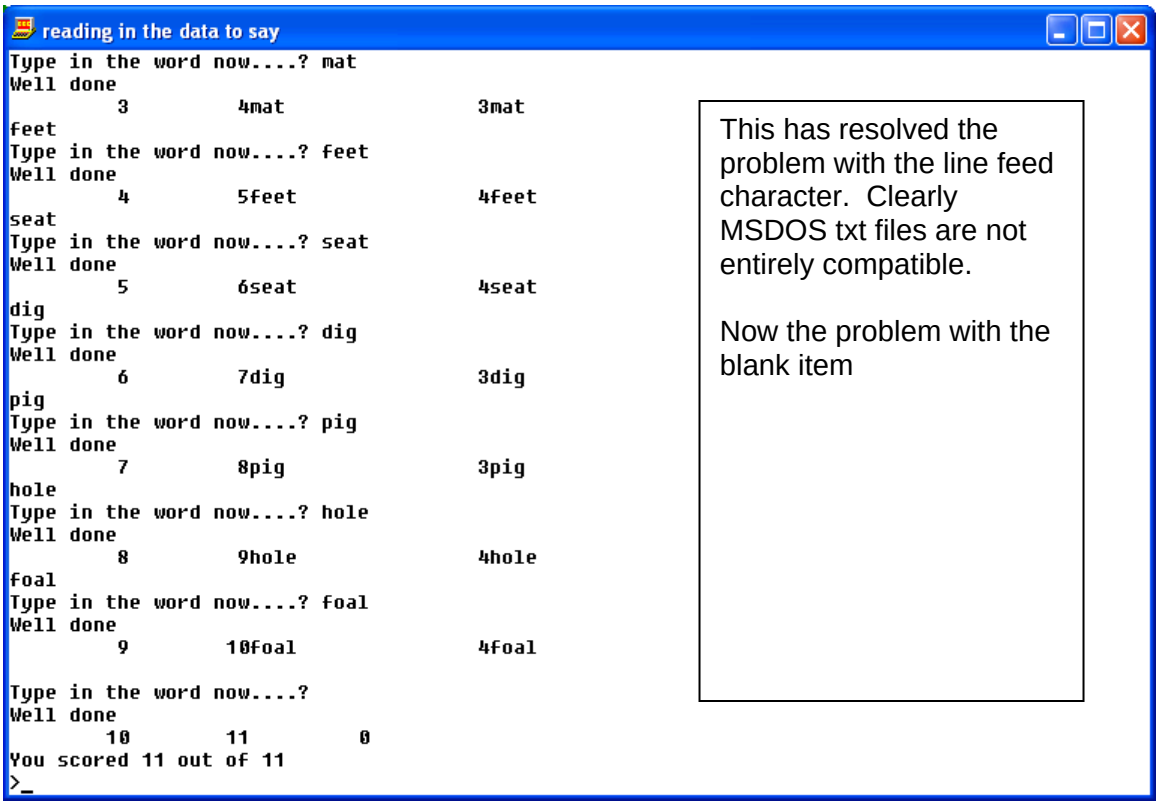
Create an array to input data into
Input 10 items of data

Open a file for output

Print the data to the file

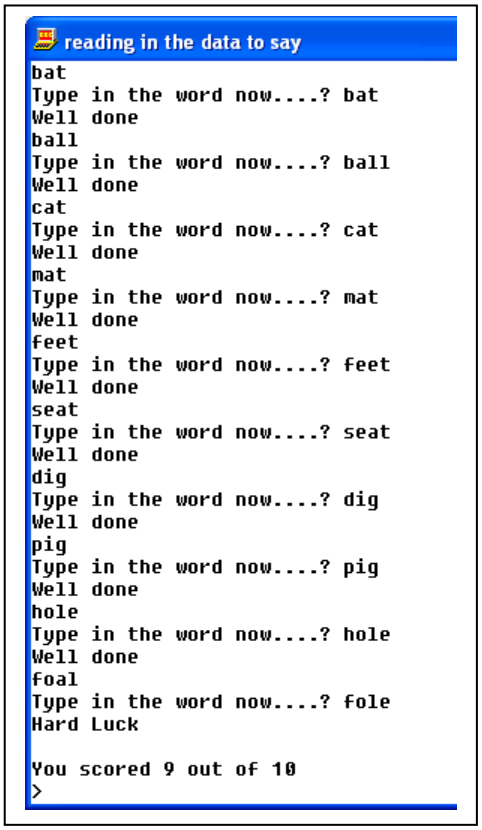
Close the file

Now I have modified my code to take data from this file rather than the one created with notepad.



This has resolved the problem with the line feed character. Clearly MSDOS txt files are not entirely compatible.

Now the problem with the blank item

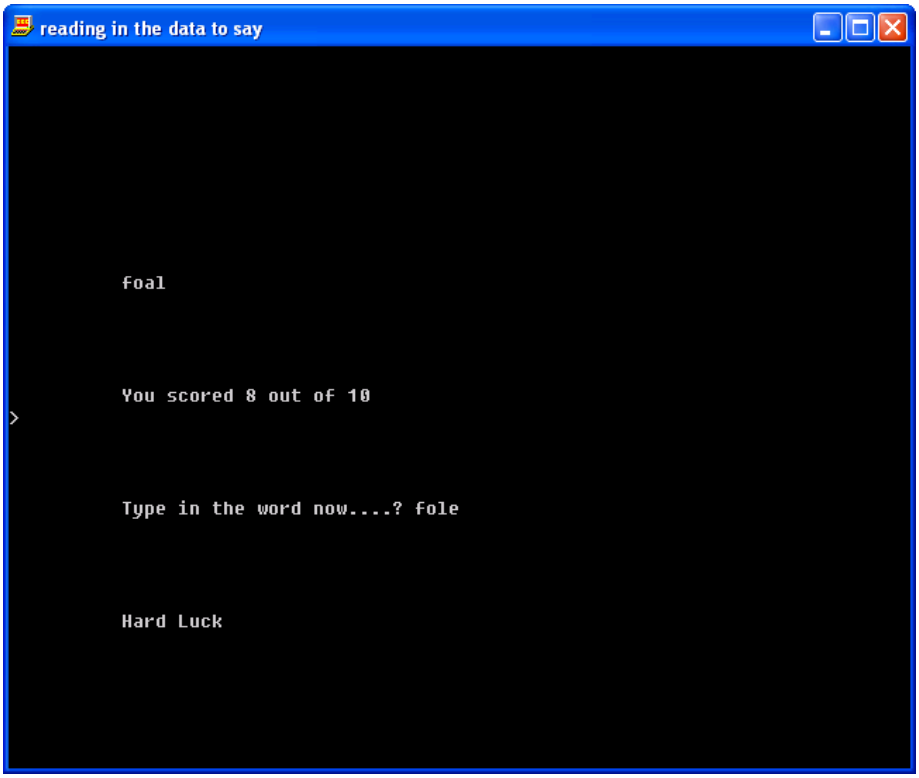



```
  DIM item$(10)
DIM student$(10)
correct=0
a=OPENIN "C:\data.txt"
FOR j=0 TO 10
  INPUT#a,item$(j)
NEXTj
CLOSE#a
i=0
REPEAT
  word$ =item$(i)
  PRINT word$
  IF word$="" THEN
    PRINT "You scored ";correct;" out of "i
  ENDIF

  PROCspeak(word$,Pitch%,Speed%,Voice$)
  IF word$<>"" THEN
    INPUT "Type in the word now....",student$(i)
    IF student$(i)=item$(i)  THEN
      correct=correct+1
      PRINT "Well done"
    ELSE
      PRINT "Hard Luck"
    ENDIF
  ENDIF
  i=i+1

UNTIL word$ = ""
```

This code now works by checking for the last blank item before outputting the final score and only printing and saying the word if it is not blank

Now I need to place the printed text at fixed locations so that it overwrites itself each time.

By using the PRINT TAB(x,y) function I was able to position the text on a graphics screen. This is MODE 0 and probably too fine for this program, I will need to ensure the spaces are cleared before writing new words on screen and experiment with the different screen modes.



Using a simple routine to blank out previous screen output before printing and saying the word, and before asking for input the screen layout now functions correctly.

```
  FOR k=0TO 15
  PRINT TAB(k+5,8) " "
NEXT k
PRINT TAB(5,8) word$
IF word$="" THEN
  PRINT TAB( 5,12) "You scored ";correct;" out of ";i
ENDIF
FOR k=0TO 14
  PRINT TAB(k+28,16) " "
NEXT k
```

This routine simply replaces what is on screen with blank spaces.

NEXT some colour has been added to the text using the colour statement. Red for hard luck, yellow for key words and well done, white for line where response is typed..

I also tried out different background colours, the grey appeared to be easiest to read.

The main body of the program now works and I need to work on choosing word lists and entering word lists. I have currently set the limit to 10 but this can easily be modified at the implementation stage.

My routine to enter and save data to a file has already been developed and the simplest solution for the desired three word lists will be to CHAIN this routine from a menu for each of the three files.

To select a word list all I need to do is to load the appropriate file from a simple menu.

This is the code I have written to use the pre written routines for the speech module.

```
MODE 19
COLOUR 136
CLS
DIM item$(10)
DIM student$(10)
correct=0
a=OPENIN "C:\data.txt"
FOR j=0 TO 10
  INPUT#a,item$(j)
NEXTj
CLOSE#a
i=0
REPEAT
  word$ =item$(i)
  FOR k=0TO 15
    PRINT TAB(k+5,8) " "
  NEXT k
  COLOUR 3
  PRINT TAB(5,8) word$
  IF word$="" THEN
    COLOUR 3
    PRINT TAB( 5,12) "You scored ";correct;" out of ";i
    PRINT TAB(3,16) "                                    "
  ENDIF
  FOR k=0TO 14
          PRINT TAB(k+28,16) " "
```

Set the screen mode, background colour and clear the screen.

Dimension the arrays for the word list and student input

Initialise a variable to store the current score

Open a file for input and
Input the data from file into an array

Close the file
Initialise a loop variable

Loop
Select word from list

Clear the area on the screen

Set the text colour and
Print the word on screen

If the word is blank then
Set the text colour and
Print the final score on screen

```
  NEXT k

  PROCspeak(word$,Pitch%,Speed%,Voice$)
  IF word$<>"" THEN
    COLOUR 7
    INPUT TAB(3,16) "Type in the word here..",student$(i)
    IF student$(i)=item$(i)  THEN
      correct=correct+1
      COLOUR 6
      PRINT TAB(5, 20) "Well done"
    ELSE
      COLOUR 1
      PRINT TAB(5,20) "Hard Luck"
    ENDIF
  ENDIF
  i=i+1

UNTIL word$ = ""

PROC_comexit
END
```

Clear the area for input

Call the speech routine with the current word
IF the word is not blank then
Set the text colour

Print instructions to type in word

Compare input to current word and update score accordingly

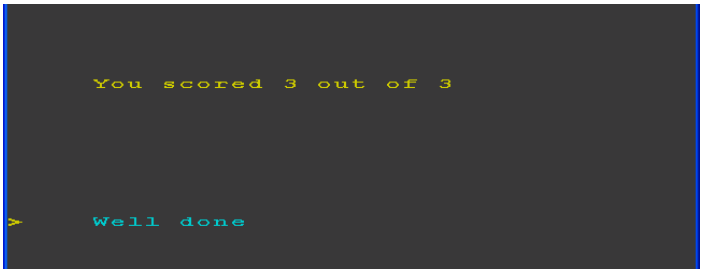
Set the text colour accordingly and print well done or hard luck

Increment counter and repeat the loop until the word selected is blank.

I have tested this routine with some sample data as I developed it. The results are shown alongside the development.

The test data had 10 items, the maximum allowed by the dimensioning (this can easily be altered) but it needs to be tested with data files containing fewer items than the maximum allowed.

I have created a data file with 3 items, cat, bat and ball.



This works without any problems

A file with non –words should be spoken but be meaningless

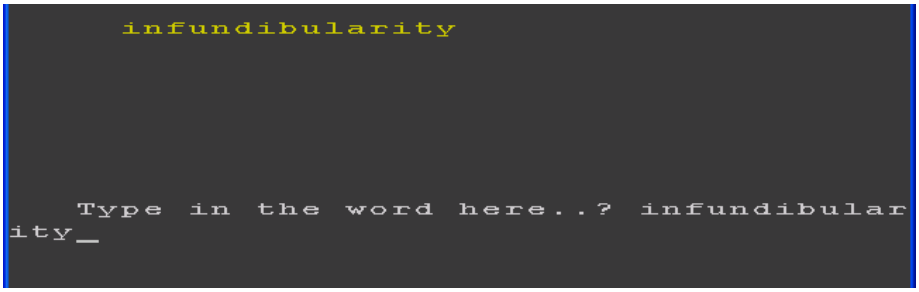
My file is ghjk, namadfad, 2e3rtg*

As suspected this causes no issue, the spoken output is just meaningless but the system functions normally.

The system is designed for primary school and short words, there is a maximum text screen size of 40x30 so words longer than 12 will wrap around.

I have used the following words to show this.

Infundibularity, functionality, myristicivore



The word wraps around when typed in but the system continues to work correctly.

The system could easily be modified if this represented any problem.

To write data to files I have created a simple program:

```
INPUT "File name " file$
filename$="C:/"+file$+".txt"
INPUT "How many words? "size
DIM word$(size)
FOR x= 1TO size
  INPUT "word ", word$(x)
NEXT
a=OPENOUT filename$
FOR i=1 TO size
  PRINT#a,word$(i)
NEXT
CLOSE#a
```

Request a name for the file
Set filename to point to text file on C drive with .txt extension

Ask for number of words
Dimension array to number of words

Get input from user

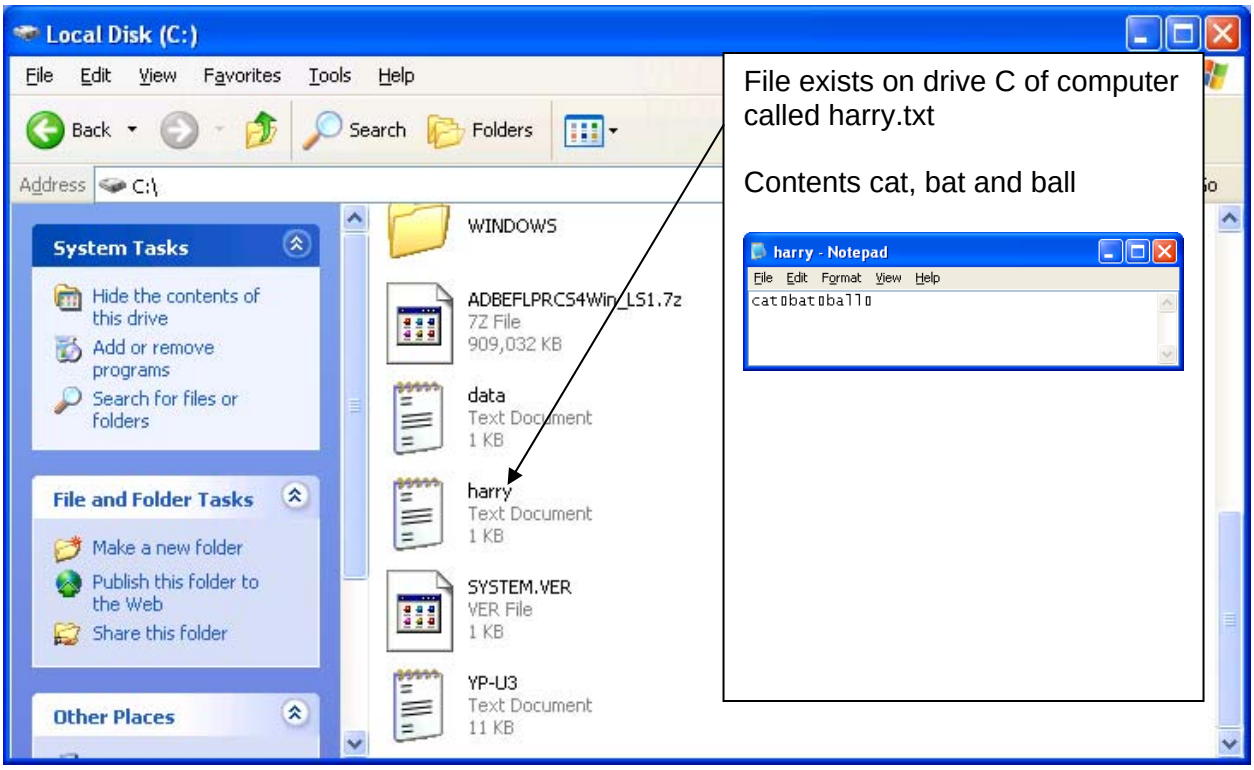
Open the file with supplied name for output

Write data to file

Close the file

This will allow the teacher to create any number of files of any size for use with the student.

Test: file name harry, 3 words, cat, bat and ball.



Now a simple modification to the start of the main program allows the user to select a data file.

```
MODE 19
COLOUR 136
CLS
PRINT TAB(3,5) "Which data file?"
INPUT TAB(5,8) file$
filename$="C:/"+file$+".txt"
CLS
DIM item$(100)
DIM student$(100)
correct=0
a=OPENIN filename$
```

Set mode, colour and text colour

Clear the screen
Ask for filename

Use data entered to create full pathname to file

Clear screen

Dimensioned to allow for large numbers of words

Open the file using the filename variable

The main program is set to store a fixed amount of data in an array. If I store the size of the data file as the first item in the data file itself I will be able to retrieve this and dimension the main program variables to the correct size.

A simple modification to the write data program allows the size to be stored as the first data item.

```
INPUT "File name " file$
    filename$="C:/"+file$+".txt"
    INPUT "How many words? "size
    DIM word$(size)
    FOR x= 1TO size
      INPUT "word ", word$(x)
    NEXT
    a=OPENOUT filename$
    PRINT#a,size
    FOR i=1 TO size
      PRINT#a,word$(i)
    NEXT
    CLOSE#a
```

Some modifications to the speak words program allows the dimensions to be set to the correct value

```
a=OPENIN filename$
INPUT#a,size
DIM item$(size)
DIM student$(size)

FOR j=0 TO size
  INPUT#a,item$(j)
NEXTj
CLOSE#a
```

Now a simple menu system to link all the components together:

Menu options become:
1. Take test
2. Create word list
3. Quit

(Using a filename for the data file allows me to enable as many tests as the teacher wants rather than the original idea of three set tests to be modified)

```
   REM Menu
 20 MODE 19
   COLOUR 133
   CLS
   PRINT TAB(5,5) "Spelling Test"
   PRINT TAB(3,8) "1.  Take Test"
   PRINT TAB(3.10) "2.  Create a new word file"
   PRINT TAB(3.12) "3.  Quit"

   g=GET
   PRINT g

   IF g<49 OR g>51 THEN
     CLS
     PRINT "select a number between 1 and 3 please"
     GOTO 20
   ELSE
     IF g=49 THEN
       CHAIN"Reading in the data to say"
     ELSEIF g=50 THEN
        CHAIN"Writing data to file"
       ELSEIF g=51 THEN
          END
        ENDIF
       ENDIF
     ENDIF
   ENDIF
```

Set mode and colours

Clear screen

Print menu options on screen

Wait for key press and take ascii value of key pressed

If not 1,2 or 3 then clear the screen
ask for a correct value and

go back to the menu

If 1 then start the take test program

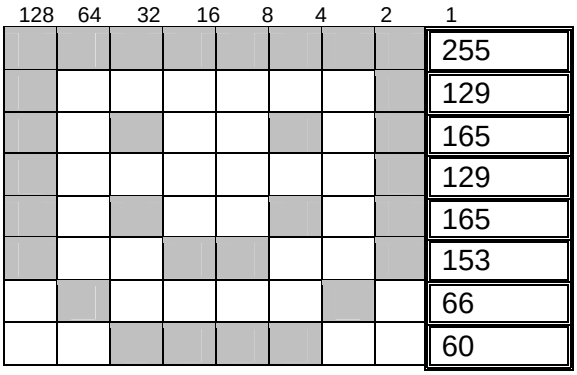
If 2 then start the enter words program

If 3 then end

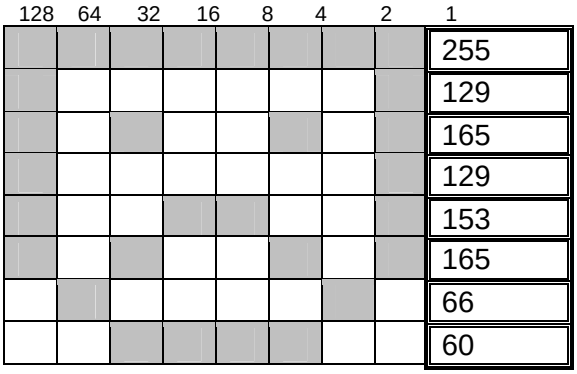
I will now need to replace the ‘well done’ and ‘hard luck’ with suitable characters.

BBC basic has a VDU command, VDU 23 which allows the user to define new characters using an 8 by 8 grid

A smiley face:



A sad face



In the code the vdu defines the characters and a string variable is used to create a line of the appropriate faces to display

```
VDU23,130,255,129,165,129,165,153,66,60
VDU23,131,255,129,165,129,153,165,66,60
smile$=CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)+CHR$(130)
sad$=CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)
```

smile$ or sad$ displayed depending upon the input.

```
 IF student$(i)=item$(i)  THEN
  correct=correct+1
  COLOUR 6
  PRINT TAB(5, 20) smile$
ELSE
  COLOUR 2
  PRINT TAB(5,20) sad$
ENDIF
```

To store the data about which words proved difficult for the student I have included a request in the code before selecting the test to take the student's name.

This name is used to create a file on the chosen drive (set to C for test purposes).

I have also created an array called failed$ the same size as the words list.

Using the following code

```
  IF student$(i)=item$(i)  THEN
 correct=correct+1
 COLOUR 6
 PRINT TAB(5, 20) smile$
ELSE
 COLOUR 2
 PRINT TAB(5,20) sad$
 fail=fail+1
 failed$(fail)=word$
ENDIF
```

I can write the words that proved difficult from the array

And with this code the data is transferred from the array into the file

```
  incorrectfile$="C:/"+studentname$+".txt"
b=OPENOUT incorrectfile$
FOR count=1 TO fail
  PRINT#b,failed$(count)
NEXT count
CLOSE#b
```

I just need the menu option to open this file and display the contents so that the teacher can identify which words have caused a problem.

```
  MODE 19
COLOUR 134
COLOUR 1
CLS
INPUT "File name " file$
filename$="C:/"+file$+".txt"
a=OPENIN filename$
REPEAT
  INPUT#a,word$
  PRINT word$
UNTIL EOF#a
CLOSE#a
```

```
150 MODE 19
   COLOUR 136
   CLS
   PRINT TAB(3,5) "Which data file?"

   INPUT TAB(5,8) file$
   filename$="C:/"+file$+".txt"
   CLS
   correct=0
   a=OPENIN filename$
   IF a=0 THEN
     PRINT TAB(5,5)"Data file does not exist"
     WAIT 300
     GOTO 150
```

I have included of a label to return control to this point

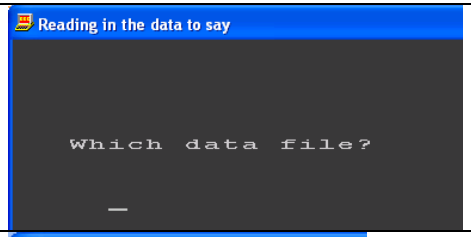
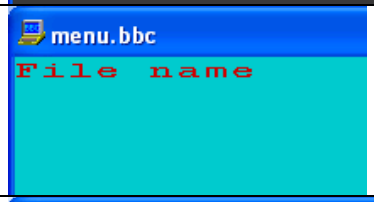
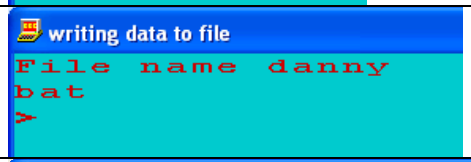
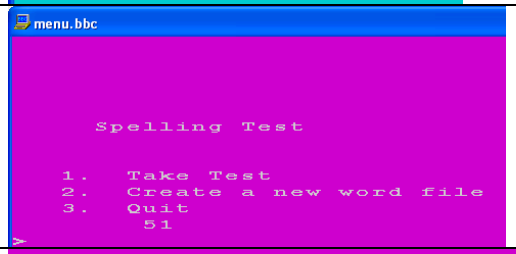
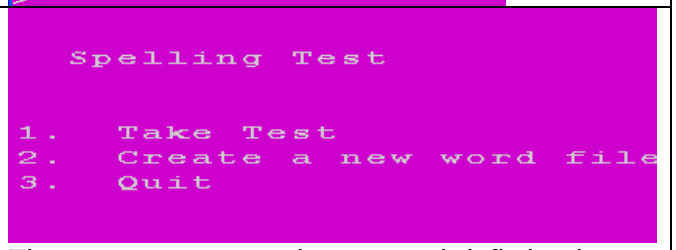
A check on the file channel value to see if the file exists

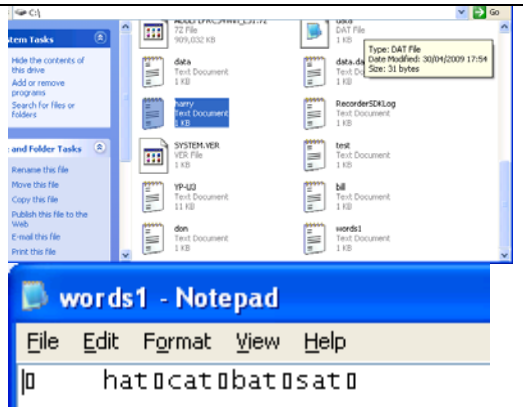
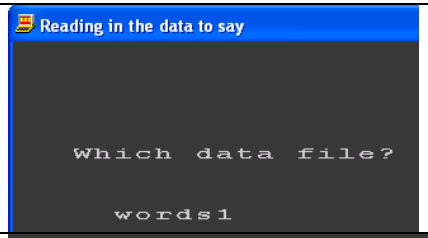
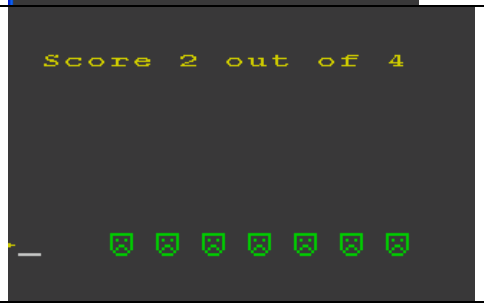
Wait for 3 seconds for the user to read the error message before returning the control to the point marked by the label.

```
   IF g<49 OR g>51 THEN
     CLS
     PRINT "select a number between 1 and 3 please"
     WAIT 300
     GOTO 20
```

A wait command has been included to give the user time to read the error message

Test plan for full system:

Test	Data	Reason	Expected	Actual
Menu system	1	To see if take test program start	The screen will clear and it will ask me which test I want to take	
	2	To see if create new word list program starts	The screen will clear and it will ask me for a file name	
	3,danny	To see if the student incorrect responses can be read	The screen will clear and display the contents of the student file saved as danny	
	4	To see if the system quits cleanly	The system should stop	
	5,8, -1,0 a,	Invalid data	An error message will appear and it will return to the menu	 The error message only appears briefly but it does return to the main menu. I will add a delay to show the error message.

Create a word list	Words1, 4, cat, hat, bat, sat	Valid data,	Should ask for filename, size and 4 data items and create a file called words1 on the C drive	
Take a test	Select test words1	Valid file name	Should load the test and ask me to spell 4 words	
				
	Select file noexist	File does not exist.	Should fail and give error message.	Message invalid channel and program stops. I need to add an error trapping routine to reject false file names and return to the file name request

My system design was to create a programmed solution using BBC Basic as a spelling aid for primary school children with the following features:

• A simple menu interface with 4 choices: o Take Test o Create word list o View student file o Quit	I have created a simple functional menu with these choices. The developed system can handle any number of tests. The take test, create test, view data and quit options work effectively
• The test interface keeps a score	The score in the test is displayed correctly at the end of the test
• The word may be displayed for the student to copy	The word is displayed, with * to represent the missing characters to be overtyped.
• Simple space to type in the word	The word is typed over a prompt on screen,
• The system speaks the word using a computer voice	I found a module by R T Russell that I have used to speak the words from a file
• Success or failure indicated	The system displays a set of smiley or sad faces according to the success or otherwise

I asked my classmates to try out the system. Their comments include.

1. The computer voice isn't always clear
2. The words are not really large enough
3. The smiley faces look creepy, why not use a tick or cross instead?
4. Entering wordlists is dead easy
5. Like the way the incorrect words are kept for the teacher
6. Great but why not just copy words on a paper worksheet; learn to write then as well.

My comments on these:

1. I can play with the sound parameters to improve the voice so some work on that will deal with this criticism.
2. The mode I chose was 19 because of the number of colours, I didn't use them all so I could use a different mode with larger text, eg mode 5, OR I could create a simple character set using the VDU commands. The second option would produce a better result.
3. Agree, why didn't I think of that? Two blocks to create a large tick using the VDU23 command.
4. Positive
5. Positive
6. Yes, it does sound just as good, perhaps the motivational aspects of using a computer are worth the extra effort, or perhaps the writing skills are a separate issue.

The initial group sessions identified some basic concepts but not the fact that these children would not be able to read the instructions. The discussions did not deal with the fact that a demonstration program or a simple flash presentation would probably be as useful for this level of activity and spelling programs for older children might have been more useful. The feedback from the other students was useful at the end of the process and helped me refine some issues and identify things to develop.

End of candidate style response

Notes from examiner

There is a basic analysis of the problem with some evidence of planned information gathering related to the initial problem. There is some reasonable research into existing solutions covering a number of aspects of the problem. There is a clear attempt to link the analysis to the end user requirements and this leads to useable design requirements. There is some thought given to the software and hardware requirements.

Analysis mark:
0-3 All the items in this category are covered.
4-7 There is evidence of planning and an attempt to link this to the problem requirements. Some success criteria are given and a limited analysis of existing solutions.
8-10 There is evidence of design requirements and some limited justification for the design.
Overall 8/10 for analysis

There is a set of clear algorithms that provide a solution to the stated problem and a set of screen designs. There is also some discussion of how the proposal solves the problem. There is a clear test strategy and detailed test plan.

0-4 All easily covered apart from the test strategy
5-8 Good algorithms and designs
9-12 Algorithm detailed enough as are the designs
Overall 9/10 for the design because of the algorithm

Standard programming constructs are used effectively in the solution and there is good evidence of suitable select statements, loop structures, file handling etc. Variable names are not meaningful in many cases but at least these are not ambiguous. Standard use of i, j and k as loop indices is acceptable. Arrays are used effectively. Solutions are effective and fully functional but not always efficient. This is work bordering on 9-11 with few inadequacies.
Overall 9/11 for use of coding features

The development is very clearly described with excellent evidence to show the development of the solution. There is a critical discussion of the process showing how the plan has been modified in light of testing during development. The commentary is clear and detailed and the overall solution is effective. There are some very minor issues and inefficiencies but no significant errors hence 6-7 marks
Overall 6/7 for development

There is a test plan and the development testing is well structured and clear, the final product testing is planned and covers various aspects of the system. There is some evidence of others testing the system and the developer has considered issues with implementing on systems other than the development computer - hence 8-10 category.
Overall 8/10 for testing

There is an evaluation of the system against the design criteria. Modifications and limitations have been identified and there is evidence that these have been dealt with.
There is evidence of comment on group activities and a response to these comments.
The report is well structured with good use of technical language and few errors in punctuation, grammar and spelling.

Very few omissions making this top range work.
Overall 8/10 for evaluation

Overall mark 48/60 (approximately A grade)

We would expect this to be supported by more evidence than can be seen in the report. There should be a final listing for all the component programs with screen shots of the system being used. (For brevity this evidence has not been included.)

This programming language allows the programs to be compiled to produce stand-alone executables. The candidate should submit these files together with some research, a design specification, a test plan and conclusions rather than trying to provide all the evidence in printed format.