PRIFYSGOL
BANGOR
UNIVERSITY

Imperative Programming in C

Assignment: Word Search

Dr. Llyr ap Cenydd

Introduction

A Word Search Puzzle consists of a grid of letters, with words hidden inside horizontally, vertically and diagonally. The player's task is to find all the words hidden in the grid.

Your task for this assignment is to create a Word Search Puzzle generator, and a program that lets the user play the game against the clock.

An example Word Search Puzzle can be seen below.

CAREERS WORD SEARCH PUZZLE

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

ACCOUNTANT	ENGINEER	PHARMACIST	PROGRAMMER
CARPENTER	FIREFIGHTER	PHYSICIAN	REPORTER
COURT REPORTER	LIBRARIAN	POLICE OFFICER	SURVEYOR
DISC JOCKEY	MECHANIC	POLITICIAN	VETERINARIAN
ELECTRICIAN	MUSICIAN	PROFESSOR	WEB DESIGNER

www.WordSearchAddict.com

In our version, we will ask the user to pick a category from a list, create a word search puzzle based on that category and then let them type words they can see in the grid. If they find all words before a timer reaches zero, they win!

Specification

Your program should have the following features (**30 marks in total**):

1. Word Search Generation

a) Store a list of words on specific subjects (animals, colors, planets etc.) **(2 marks)**

b) Generate a 2D grid of letters, containing at least 6 words from a category found in (1a). The size of the 2D grid should be definable (minimum 12x12).

- The words should be randomly placed within the grid
  i. Horizontally (40% of the time) **(2 marks)**
  ii. Vertically (40% of the time) **(2 marks)**
  iii. Diagonally (20% of the time) **(4 marks)**

  Diagonal words can be ascending or descending.

c) Ensure words do not overwrite one another. **(5 marks)**

2. Word Search Game

a) Welcome user to the program. **(1 mark)**
b) Present the user with a choice of categories from (1a). **(1 mark)**

c) Ask user to type in words that they can see in the grid **(2 marks)**
d) Highlight or remove correctly guessed words from the grid **(4 marks)**

e) Show message to user if time limit is reached, or all words found **(2 marks)**

For (2e). You do not need to end the game exactly when timer reaches 0, only when user types in a word after timer has reached 0.

Coding

- Programming Proficiency **(3 marks)**
- Commenting / Formatting **(2 marks)**

Programming proficiency refers to how well your program runs, how it looks etc. Marks are awarded for impressing the assessor here.

Commenting / Formatting marks are awarded for clear documentation of your code through appropriate comments, correct indentation etc.

Implementation Guide

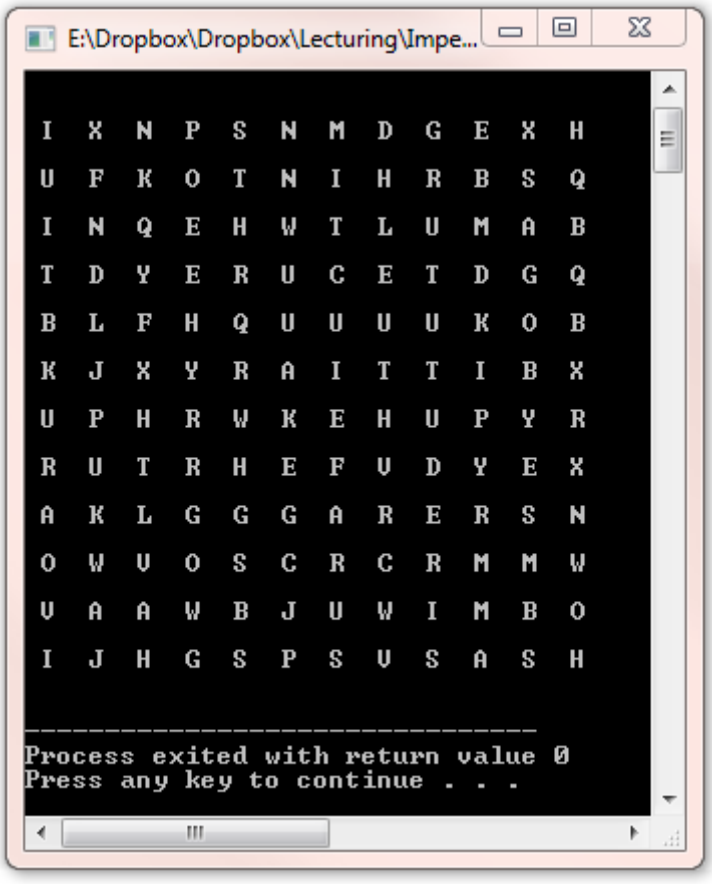
While you are free to code your program any way you want, it is recommended you follow this implementation guide especially if you are unsure what order to develop your program.

Stage 1 – Grid Initialisation

The appendix contains a shell of a program that you can use as a starting point for this assignment. The code will create a 2D character array and fill it with '.' characters.

The first stage for your assignment should be to initialise a 2D array containing a random grid of letters from the alphabet, and print the grid with appropriate formatting.

By the end of this stage your program should generate output similar to the following:



Example function prototypes for this stage are:

```
char **create2DArray();          //creates a 2D array of characters, returns pointer to array

void printArray(char** array);   //prints out a 2D array on screen
```

Stage 2 – Inserting Words Horizontally

To make testing and debugging easier you should initialise your grid with a common character for now. In the following examples, the grid is filled with '.' characters rather than random letters.

Extend your program so that it can insert horizontal words into the grid at random positions. The recommended function prototype is:

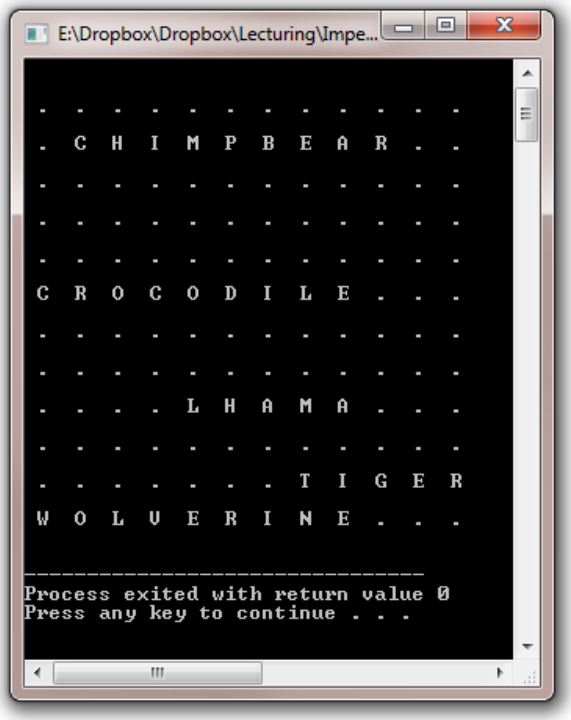
```
void insertHorizontally(char* word, char** array);
```

Eventually you will need to test that you are not overwriting words already inserted into the grid. However for the time being do not worry about testing for this.

Hints

- Look at previous labs, lectures and code snippets for examples of how to insert information into a 2D array
- Use a for-loop to insert the word into the array one character at a time
- You can use strlen() to get the length of a string.
- When inserting a word into the array, you will need to make sure that there is enough horizontal space for all the letters.

When you've finished this stage, your program should be able to pick from an array of words and place them at random horizontal positions in the array, similar to the following screenshot:



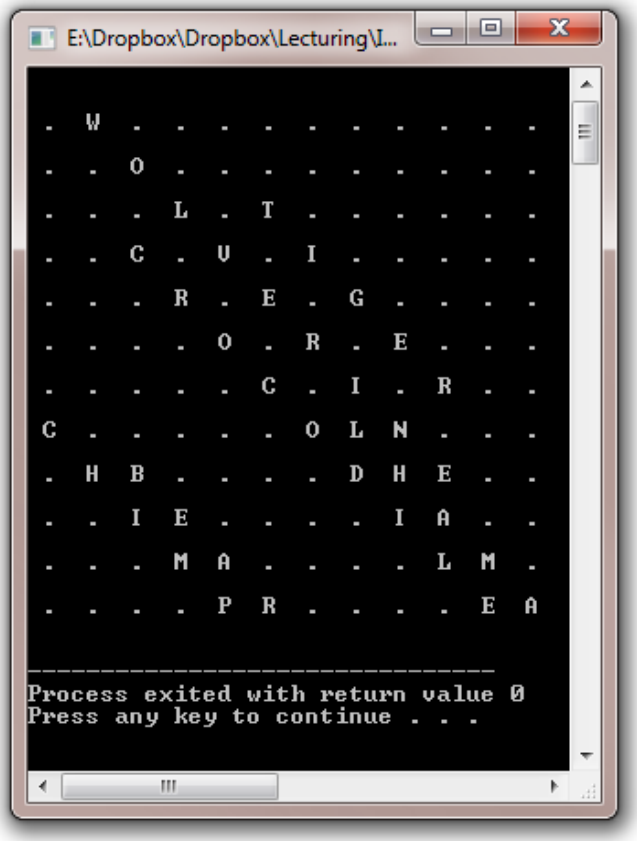
Stage 3 – Inserting Words Vertically/Diagonally

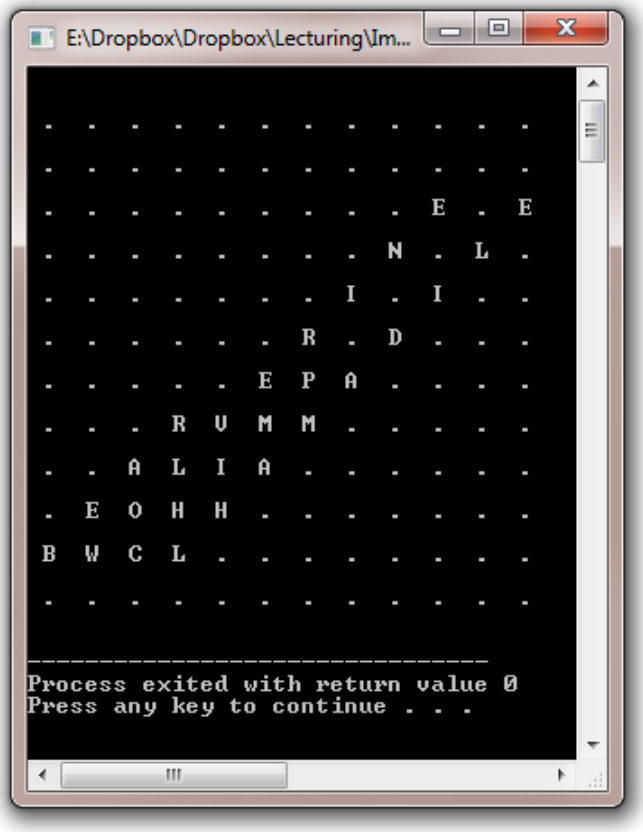
The next stage is to extend your program so that it can display words vertically and diagonally.

```
void insertVertically(char* word, char** array);

void insertDiagonally(char* word, char** array);
```

Remember that diagonal words can be ascending or descending. For example





Test your program by inserting a mixture of horizontal, vertical and diagonal words into the grid.

You might have noticed that some words overlap in the example screenshots so far (Chimpbear, Lhamadile). This is because we are not doing any testing to check if we are replacing characters that were previously placed in the grid. It is up to you if you want to work on this part first (Stage 5), or create the game first (Stage 4).

Stage 4 – Playing the game

We are now at a stage where we can write the code that plays a game using the grid and associated words.

The program should work like this:

- Welcome user to the program
- Ask the user which category of words they would like
- Generate a grid using words from this category
- Start a timer (e.g. 3 minutes)
- In a loop
    - Present the grid to the user
    - Allow the user to type in words they see in the grid
    - If they find all words in time – print congratulations message
    - If the timer reaches zero – print game over message
    - (Optional) Clear screen after every user guess
- Exit

You are free to embellish your program however you like, but you must ensure that it can perform at least the tasks above. **If in doubt refer to the specification.**

An example output of the game can be seen in the appendix.

Stage 5 – Overlapping words

You are on your own with this part of the assignment. You will need to ensure that when your program generates words to place in the grid that they do not overlap words that were previously inserted.

There are many ways to complete this stage. One potential solution is to use another 2D array which tracks where words have previously been inserted. This could then act like a mask when inserting new words.

However as previously mentioned, using a mask is just one potential solution to this stage.

General Hints

- In Windows you can clear the screen using the code system("cls");
- The code sleep(2); found in <stdio.h> will make your program pause for 2 seconds. You can use this function to manage the flow of your game.
- When seeding the random number generator using srand(), if you give it a specific number e.g. srand(5), your program will always generate the same puzzle. This can be very useful when testing your program.
- You can use strlen() to get the length of a string (found in <string.h>)
- strlen() is useful for several tasks in the assignment, such as fitting strings inside the grid, and iterating over each character in a string
- To setup a countdown timer you will need to research <time.h>. Example code can also be found in the appendix.
- The beep(x,y) function takes in a frequency and duration, and makes the speaker on the motherboard beep! For example beep(1000,1000) will beep at 1Khz for 1 second. You can use this to signify correct/incorrect messages, or even play a tune when the game is over. Windows only.
- The marking scheme is itemized, so if you get stuck on a stage or feature, consider moving on to another task.

Submission

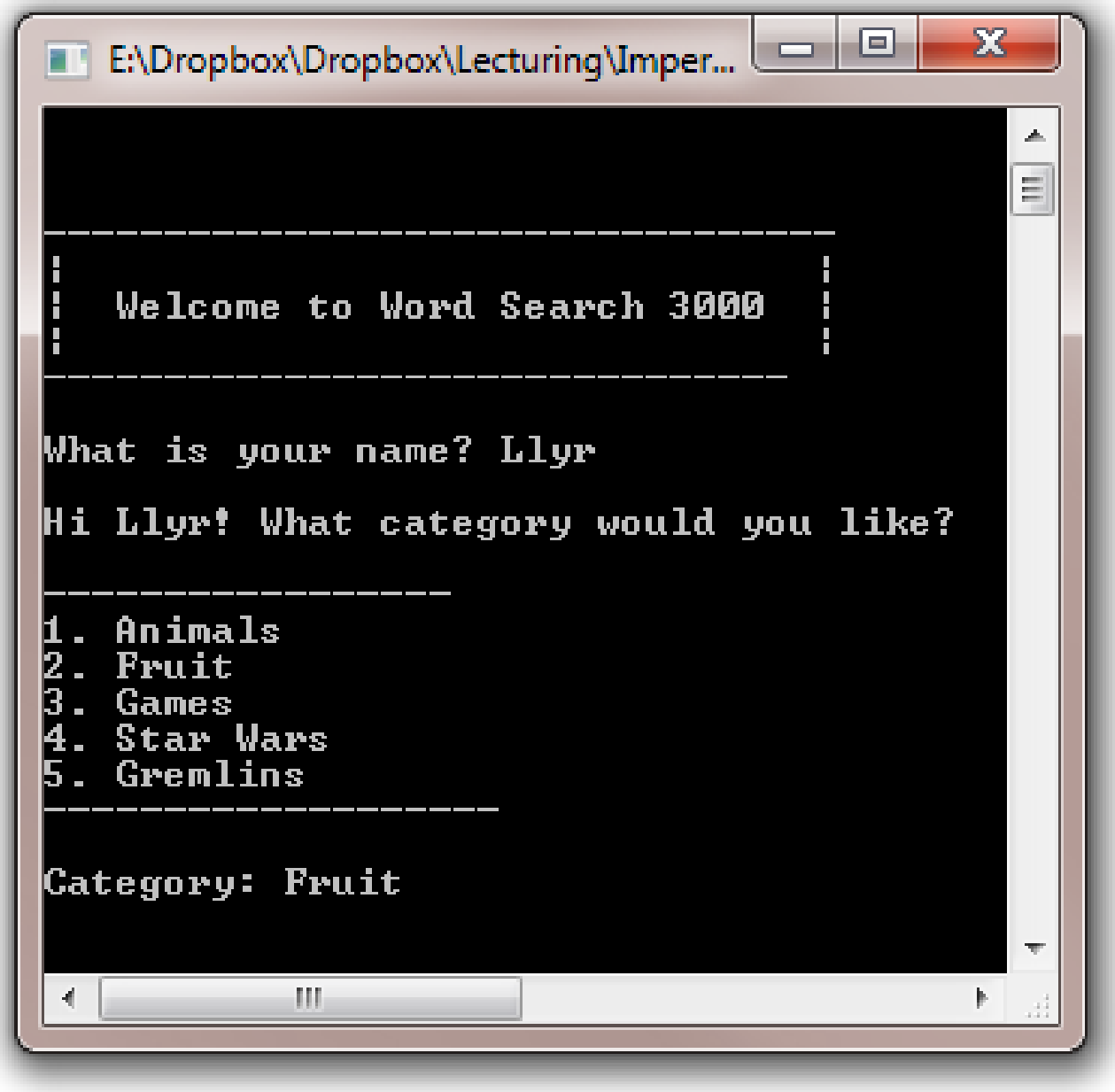
Use **Blackboard** to submit your source code for this project. You only need to upload the .c source files created for the project. Ensure that the source code:

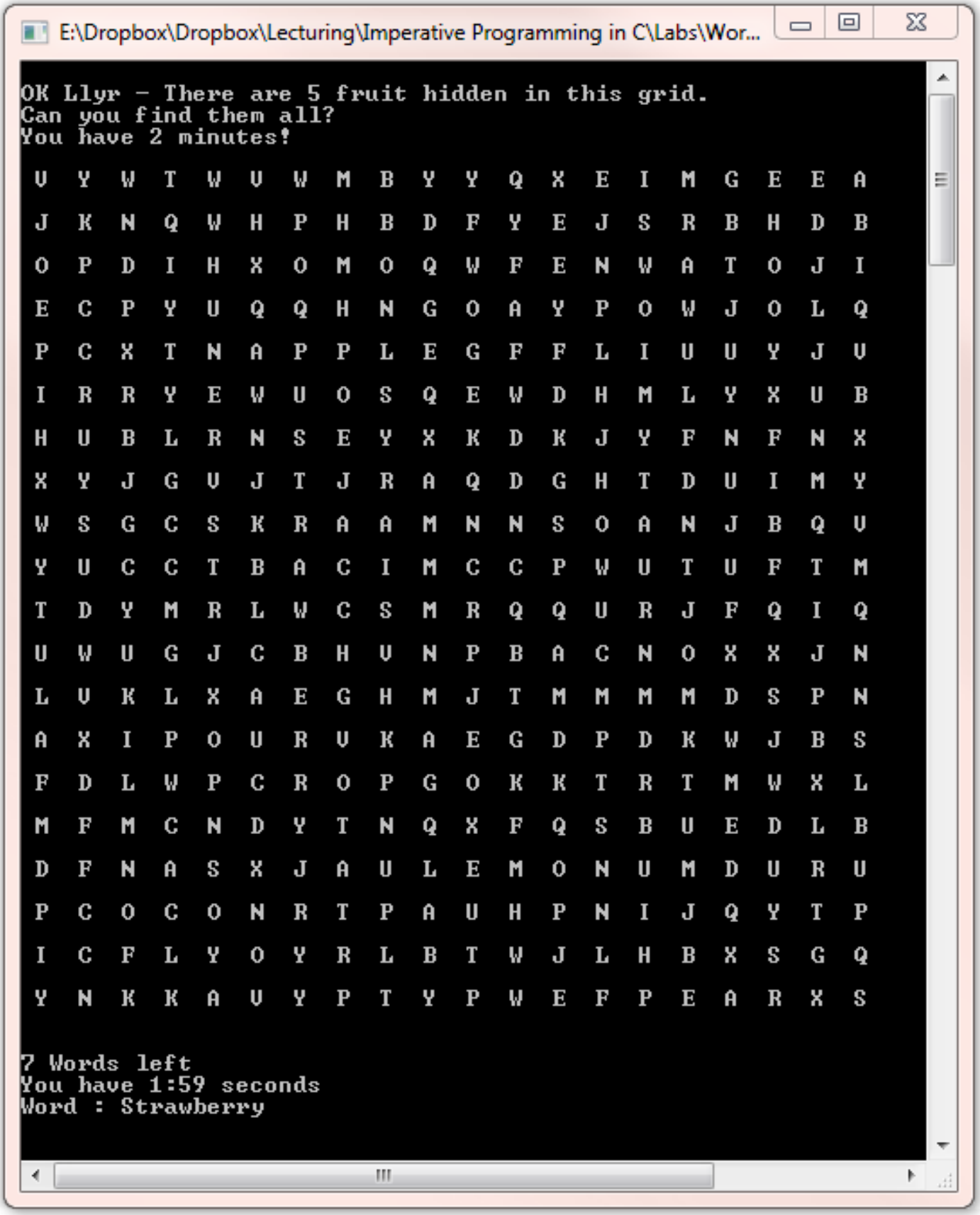
- Contains a program header
- Contains an appropriate level of comments
- Follows a consistent style of indentation
- Follows the usual C programming conventions

The deadline for submission will be published on Blackboard. Late submissions will be penalised in line with School policy.

Please note that there are severe penalties for submitting work which is not your own. If you have used code which you have found on the Internet or from any other source then you must signal that fact with appropriate program comments. You will be expected to answer detailed questions about the code you have submitted when we assess you.

Appendix A


E:\Dropbox\Dropbox\Lecturing\Imper...
Welcome to Word Search 3000
What is your name? Llyr
Hi Llyr! What category would you like?
1. Animals
2. Fruit
3. Games
4. Star Wars
5. Gremlins
Category: Fruit


E:\Dropbox\Dropbox\Lecturing\Imperative Programming in C\Labs\Wor...
OK Llyr - There are 5 fruit hidden in this grid.
Can you find them all?
You have 2 minutes!
V Y W T W U W M B Y Y Q X E I M G E E A
J K N Q W H P H B D F Y E J S R B H D B
O P D I H X O M O Q W F E N W A T O J I
E C P Y U Q Q H N G O A Y P O W J O L Q
P C X T N A P P L E G F F L I U U Y J V
I R R Y E W U O S Q E W D H M L Y X U B
H U B L R N S E Y X K D K J Y F N F N X
X Y J G V J T J R A Q D G H T D U I M Y
W S G C S K R A A M N N S O A N J B Q V
Y U C C T B A C I M C C P W U T U F T M
T D Y M R L W C S M R Q Q U R J F Q I Q
U W U G J C B H V N P B A C N O X X J N
L V K L X A E G H M J T M M M M D S P N
A X I P O U R V K A E G D P D K W J B S
F D L W P C R O P G O K K T R T M W X L
M F M C N D Y T N Q X F Q S B U E D L B
D F N A S X J A U L E M O N U M D U R U
P C O C O N R T P A U H P N I J Q Y T P
I C F L Y O Y R L B T W J L H B X S G Q
Y N K K A V Y P T Y P W E F P E A R X S
7 Words left
You have 1:59 seconds
Word : Strawberry


C:\Users\Llyr\Dropbox\Lecturing\Imperative Programming i...
OK Llyr - There are 5 fruit hidden in this grid.
Can you find them all?
You have 2 minutes!
I X T U G O P W Q R Q P E F S U N O G G
I X P I O R A N G E C G Q M I P D T M L
H V D H Q R O I D O K U G L A J P U W A
H P S P P T L C J C E C O T N P Q L E W
Q L Y D I T E W I S O H X Q M F E M U D
Q F L C J Y M T Y T O D W R I X K J Y T
C I A A W Q J G U A M P R Y G N J M U C
P R E W B H P T N P Q Y T J F O Y F D G
E X A Q N K K O J F L P Y K M J E A T M
A W X B N A X U S J W P R F C Q R P G N
R R N F Q U X G V V M F L P O J X P X G
B B E R U Q L D T B I P J A C I M L F K
X B A W X X I U J N J D Y F O Q A E W P
Y O H J V T M R J U L J P L N T V E Y H
X O O U O G E G J L E W C E U L V N L W
F W V O J K A Q K M I W B M T J A L F H
T B J L F Y A S L C X Q U O D F G O T K
P T S T R A W B E R R Y U N Q X S Y J V
O T V M A D J N R K Y F X V P C J F K H
G F Q Y J M S U S H O L M N Q B O A E D
1 Word left
You have 1:44 seconds
Word : PEAR
Congratulations Llyr! You found all the words in 16 seconds!!
********* PARTY TIME **********.

Process exited after 22.87 seconds with return value 0
Press any key to continue . . .

Appendix B

Create 2D Array

```
char **create2DArray();  //function prototype

#define WIDTH 16
#define HEIGHT 16

char** myArray;  //global array

void main()
{
	myArray = create2DArray();
}

//Creates a 2D array of WIDTH * HEIGHT and returns a pointer to it

char **create2DArray(){

	int i,j;
	char **array = (char **) malloc(sizeof(char *) * WIDTH);

	for(i=0; i<WIDTH; i++)
		array[i] = (char *) malloc(sizeof(char) * HEIGHT);

	for(i=0; i<WIDTH; i++)
		for(j=0; j<HEIGHT; j++)
	    // array[i][j] =  65 + rand() % 25;
	     array[i][j] = '.';

    return array;
}
```

Stopwatch

```
#include <time.h>

clock_t start = clock();
/*Do something*/
clock_t end = clock();
float seconds = (float)(end - start) / CLOCKS_PER_SEC;
```

Sleep

```
#include <stdio.h>

printf( "I'll be back in 10 seconds...\n\n" );
sleep(10);
printf( "I'm back!" );
getchar();
```