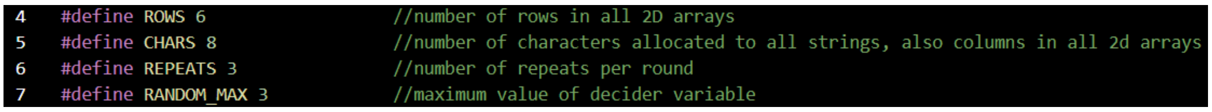
B37VB Game Project – Call and Response

James Loughlin jrgl3000@hw.ac.uk H00467477

Design

For this project the stdio and stdlib libraries were used, for input/output functions and the rand() function respectively, as well as the header file, jlgame.h. It contains the declarations of all the constants used in the main program, and the check() function which returns 0 if two strings of lengths equal to the constant CHARS are not equal to each other, and returns 1 otherwise.



```
#define ROWS 6                      //number of rows in all 2D arrays
#define CHARS 8                     //number of characters allocated to all strings, also columns in all 2d arrays
#define REPEATS 3                   //number of repeats per round
#define RANDOM_MAX 3                //maximum value of decider variable
```

Figure 1: Constants defined in jlgame.h

ROWS is equivalent to the amount of different strings used as calls and as responses. CHARS is the amount of characters allocated to all strings, including the one which stores the user's input. The value of 8 was chosen because it is the length of the longest string that would need to be stored, 'crispies'. REPEATS dictates how many times a particular call is called in a single round. Finally, RANDOM_MAX is the range of the randomly generated number which gets assigned to the integer variable, decider, and the value is 3 because that's how many sets of responses there are.

The calls are stored in an array of strings, or more accurately a 2D array of characters, named calls. There are three similar arrays of strings storing the responses, called responses1, responses2, and responses3.

In order to indicate whether the game should end due to the user having entered a wrong answer, an unsigned short integer called alive is used. It is initially set to 1 and is set to 0 if an unexpected answer is entered. The unsigned integer, cycles, is used to count the rounds played. The 1D arrays of characters, current and ans, store the selected response for a given round and user's response for a given keyboard input, respectively. The integer index is assigned a random value between 0 and ROWS-1 (equivalent to 5) at the beginning of each round and is used to address the chosen call and response within their respective arrays. Which responses array to use is dictated by the integer decider which is assigned a random value between 0 and RANDOM_MAX-1 (equivalent to 3) also at the beginning of each round. A switch statement then assigns a string to the current array accordingly.

The main program and switch statement are illustrated by the flowcharts below.

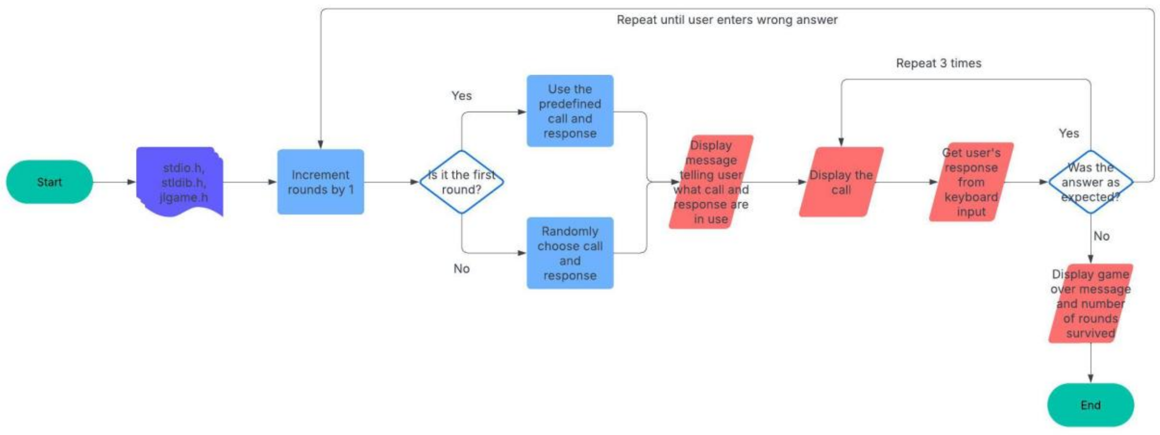

Repeat until user enters wrong answer
Repeat 3 times
Start
stdio.h, stdlib.h, jlgame.h
Increment rounds by 1
Is it the first round?
Yes
No
Use the predefined call and response
Randomly choose call and response
Display message telling user what call and response are in use
Display the call
Get user's response from keyboard input
Was the answer as expected?
Yes
No
Display game over message and number of rounds survived
End

Figure 2: Flowchart of the program

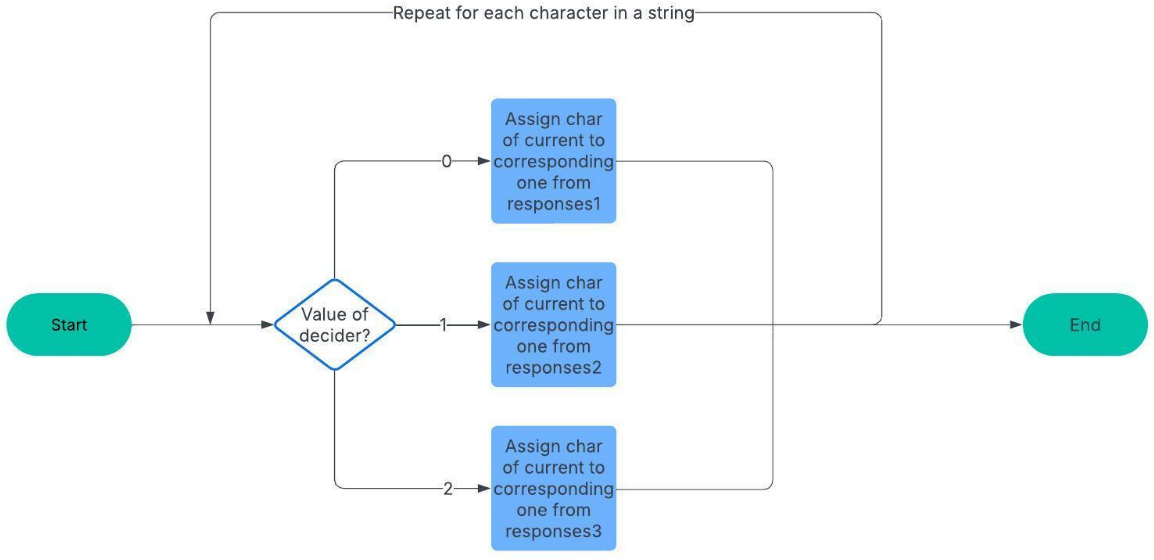
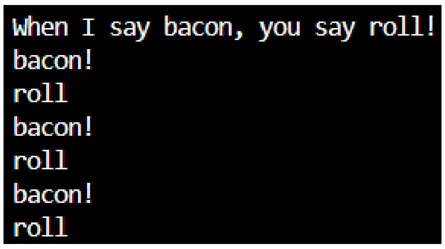

Repeat for each character in a string
Start
Value of decider?
0
1
2
Assign char of current to corresponding one from responses1
Assign char of current to corresponding one from responses2
Assign char of current to corresponding one from responses3
End

Figure 3: Flowchart of the switch statement

How to play

This game acts like having someone call a word and you respond with another word which completes a common phrase or the name of something.

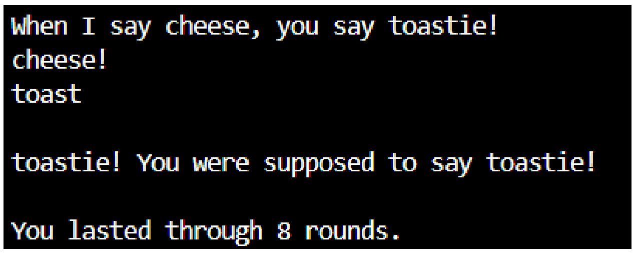
Upon starting up the program, the user is met with a message which sets out a call and the response expected. After this message, the call is printed again on a new line and the user is expected to enter the response specified. This call-response cycle is repeated three times before a new call and response are chosen at random and the next round starts. For the first round, the call and response are ‘bacon’ and ‘roll’ respectively.



```
When I say bacon, you say roll!
bacon!
roll
bacon!
roll
bacon!
roll
```

Figure 4: Call-response cycle for first round.

If an unexpected response is entered, even just a spelling mistake, the game ends and a message saying what the user should’ve said is displayed followed by another message which says how many rounds the player lasted through.



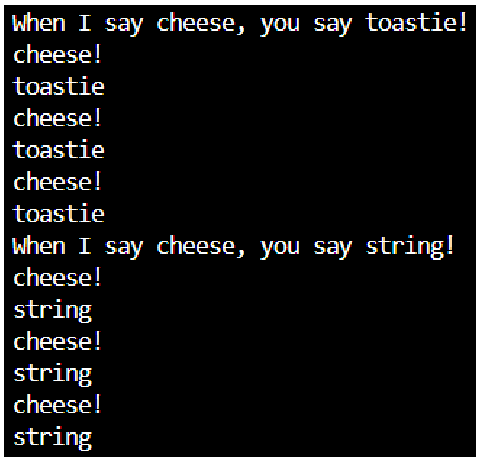
```
When I say cheese, you say toastie!
cheese!
toast

toastie! You were supposed to say toastie!

You lasted through 8 rounds.
```

Figure 5: What happens when an unexpected response is entered

Each call has three associated responses, hence the use of three parallel arrays in the implementation. Any one of the three can be chosen and it is possible for the same combination to be chosen multiple times in one game. This means the user can’t immediately know what to say from just the call alone and there are more words to keep track of.



```
When I say cheese, you say toastie!
cheese!
toastie
cheese!
toastie
cheese!
toastie
When I say cheese, you say string!
cheese!
string
cheese!
string
cheese!
string
```

Figure 6: 'cheese' being chosen as the call twice in a row, but with two different responses