|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | HangMan Flowchart  Program Design  DT2XX-1  First name Surname  Student Number : C12355251  Ronan Dillon   |  |  | | --- | --- | | *Date :* |  | | *Last Lab* |  | |  |  | | *Version:* | *1* | | *Status:* | *Draft / Release* | |  |  | |  |  | |  |  | |

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version History | | | |
| **Date** | **Version** | **Status** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |
| --- |
| Changes since last version |
|  |

|  |
| --- |
| Known Omissions |
|  |

Table of Contents

[1 The Assignment 4](#_Toc338615277)

[1.1 Overview 4](#_Toc338615278)

[1.2 The user interface 4](#_Toc338615279)

[1.3 The due date and feedback timeline 5](#_Toc338615280)

[2 Problem Definition 6](#_Toc338615281)

[2.1 Overview 6](#_Toc338615282)

[3 Design 7](#_Toc338615283)

[3.1 Overview 7](#_Toc338615284)

[3.2 Summary Flowchart 8](#_Toc338615285)

[3.3 Hangman Flowchart 9](#_Toc338615286)

[4 Test Cases 10](#_Toc338615287)

[4.1 Overview 10](#_Toc338615288)

[5 Appendix 11](#_Toc338615289)

[5.1 Appendix 1 11](#_Toc338615290)

# The Assignment

## Overview

Design a game of hangman that randomly chooses a word from a list of 20 words stored in an array of word.

Display the game as per the example screens.

There is no need to show a hang man. Instead each time a letter is picked reshow the lives left.

Display the number of characters in the word.

Ask the user to input a letter. They cannot input the same letter twice. If the letter is not part of the word they lose a life. If it is correct show the letter in its position in the word. Keep asking for letters until the user runs out of lives. At the end of the game ask if they want to play again.

The following words should be part of your 20 word list: litter, beanbag, opening, settlement.

## The user interface

Lives: 6

Word: \_ \_ \_ \_ \_ \_ \_

Letters guessed so far:

Guess a Letter:

Lives: 1

Word: P\_OG\_AM

Letters guessed so far :

A,E,I,O,U,P,G,M,Z,F

Guess a Letter :

Lives: 0

Word: P\_OG\_AM

Sorry you lost

The word was:

PROGRAM

Play again Y/N

## The due date and feedback timeline

* The assignment due date is Friday the 2nd of November.
* The assignment will be submitted in word document based on a template to be provided.
* The flowcharts submitted will be assessed in the labs over a two week period from the 5th November to the 16th November, by the lab assistants. Where the student will present their flowchart and explain it.
* The student must have their assignment with them in the lab in two forms
  + A printed copy that will be presented to the lab assistant
  + A digital copy that they can present on the computer in the lab
* They will then be marked.
* A solution flowchart will be provided once all marking is complete.
* Lab work for other students will be provided during the assessment process.
* If you are not at the lab for assessment your mark will be zero.

# Problem Definition

## Overview

Create a hangman game which asks the user to enter any character. The character that the user enters is then checked to see if it is in a secret word chosen by the program.

If the character that the user entered is in the secret word the character is displayed but if the character is not in the secret word the user loses a life. The user has 6 lives. If the player enters zero the game is exited if anything but a character is entered the user is informed that they have made an invalid entry and is asked to renter a new character.

If the player guesses all of the characters in the secret word then they are congratulated and asked if they want to play again.

If the player does not guess all of the characters in the secret word within their six lives then they are told that they have lost the game and they are asked if they wish to play again.

# Design

## Overview

*The design considerations or principles behind the design e.g. all inputs have error checking*

The design of my hangman program is as follows:

* The input made by the user will be checked to see if it is a character or the number 0. If the input is not a valid input then the user will be asked to renter.
* The program should not terminate unless the user has indicated that the wish to end the program.

**User Interface**

The Hangman Game

Lives:4

Secret word: Sheep

Characters guessed: T,A

Sorry no match. Lives=4

Make your guess!

Please enter a character

The Hangman Game

Lives: 5

Secret word:Sheep

Characters guessed: T

Sorry no match. Lives=5.

Make your guess

Please enter a character

The Hangman Game

Lives:6

Secret word: Sheep

Characters guessed:

Make your guess!

Please enter a character

The Hangman Game

Lives:1

Secret word: Sheep

Characters guessed:T,A,M,N

Sorry Invalid input

Make your guess!

Please enter a character

The Hangman Game

Lives:1

Secret word: Sheep

Characters guessed:T,A,M,N,S

Display S in secret word

Make your guess!

Please enter a character

The Hangman Game

Lives:0

Secret word: Sheep

Characters guessed:T,A,M,S,W

Sorry no match. Lives=0. You lose play again? (Y/N)

Make your guess!

Please enter a character

The Hangman Game

Lives:1

Secret word: Sheep

Characters guessed:T,A,M,N

Sorry no match. Lives=1

Make your guess!

Please enter a character

The Hangman Game

Lives:2

Secret word: Sheep

Characters guessed:T,A,M,N

Sorry no match. Lives=2

Make your guess!

Please enter a character

The Hangman Game

Lives:3

Secret word: Sheep

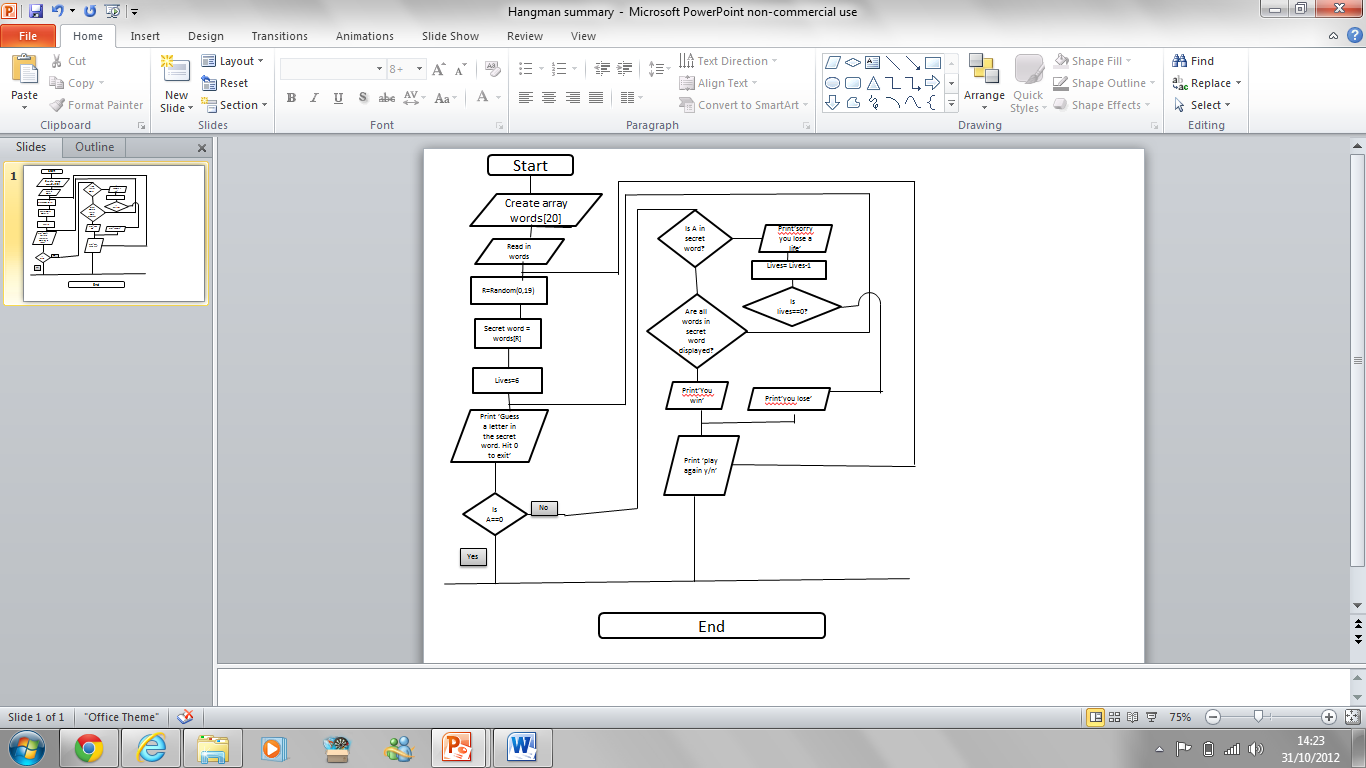
Characters guessed:T,A,M

Sorry no match. Lives=3

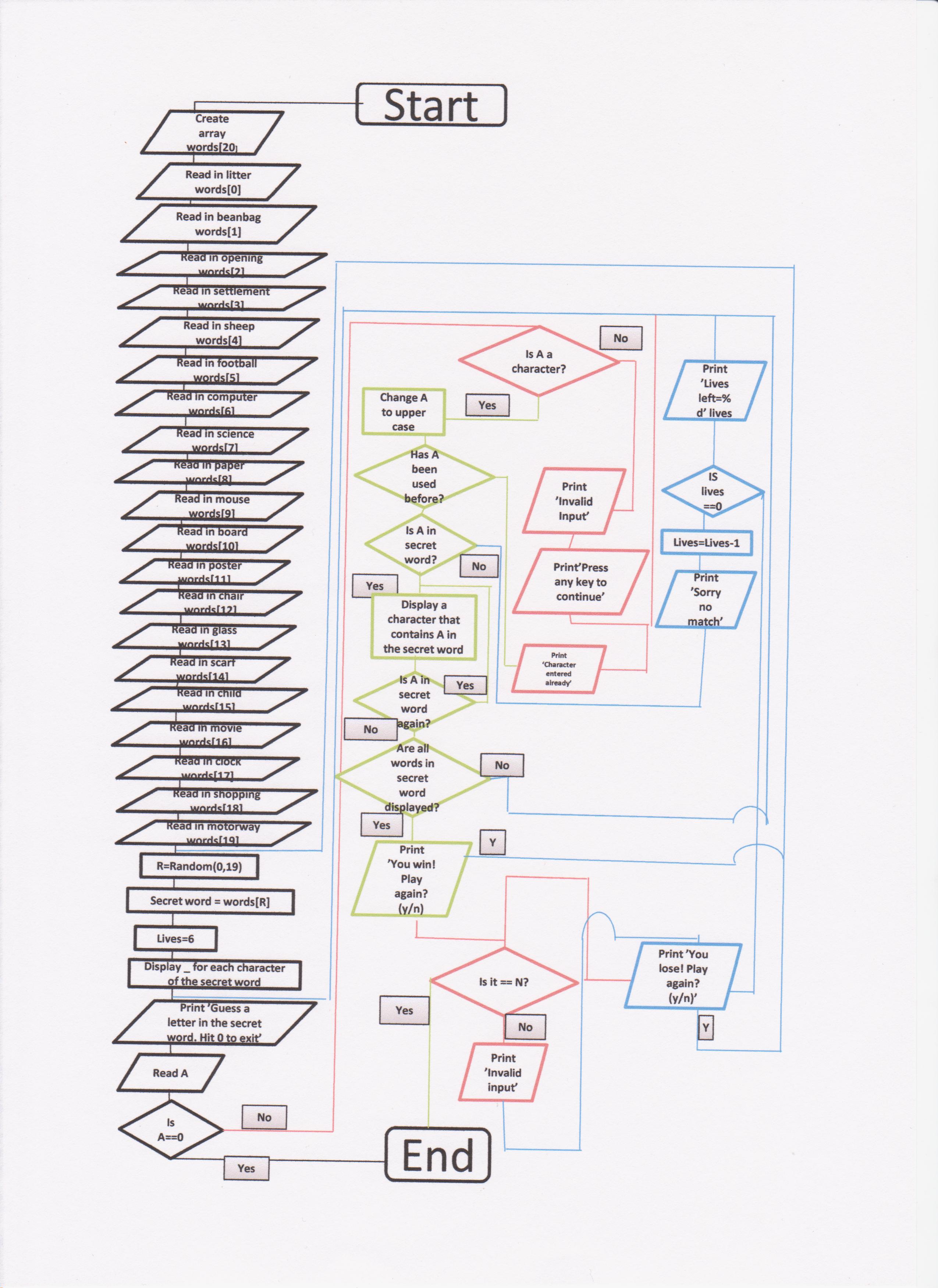
Make your guess!

Please enter a character

## Summary Flowchart



## Hangman Flowchart



# Test Cases

# Overview

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref** | **Description** | **Test case type \*** | **Expected Result** |
| TC-0010 | When the program starts the user enters 0 | Simple | The program will end. |
| TC-0020 | The user enters a number | Simple | The program will print invalid entry and ask again for another entry. |
| TC-0030 | The user enters A | Edge | The program checks if A is in the secret word if it is A is displayed if it is not lives is decreased by one. |
| TC-0040 | The user enters a | Edge | The program converts a into A and checks if A is in the secret word if it is A is displayed if it is not lives is decreased by one. |
| TC-0050 | The user runs out of lives.  The user chooses Y to replay | Normal | The user sees ‘You lost play again? (y/n)’ on the screen.  After the user enters Y the game restarts randomly choosing another word. |
| TC-0060 | The user runs out of lives.  The user chooses N to replay. | Normal | The user sees ‘You lost play again? (y/n)’ on the screen.  After the user enters N the program closes . |
| TC-0070 | The user guesses all the letters in the secret word.  The user then enters Y to replay. | Normal | The user sees ‘You win play again? (y/n)’ on the screen.  After the user enters Y the program restarts randomly chooses another word. |
| TC-0080 | The user guesses all the letters in the secret word.  The user then enters N to replay. | Normal | The user sees ‘You win play again? (y/n)’ on the screen.  After the user enters N the program closes. |
| TC-0090 | Add a log to the program so that the random number chosen by the program, the characters entered, the lives counter and the game result can be recorded. | Complex | Review the log so that you can ensure that the program is running the way you have created it to. |

\* Simple, Normal, Complex, edge condition

*You must have tests for the following words: litter, beanbag, opening, settlement – what are you testing in relation to these words?*

# Appendix

## Appendix 1