AS 91372 Assessment

Purpose

The purpose of the brief is to construct a program for the game 'Double Hog', allowing for gameplay between 2 to 4 players, following the rules as defined in the brief.

Target Audience

The target audience for this program is anyone wishing to play the game, within groups of 2 - 4.

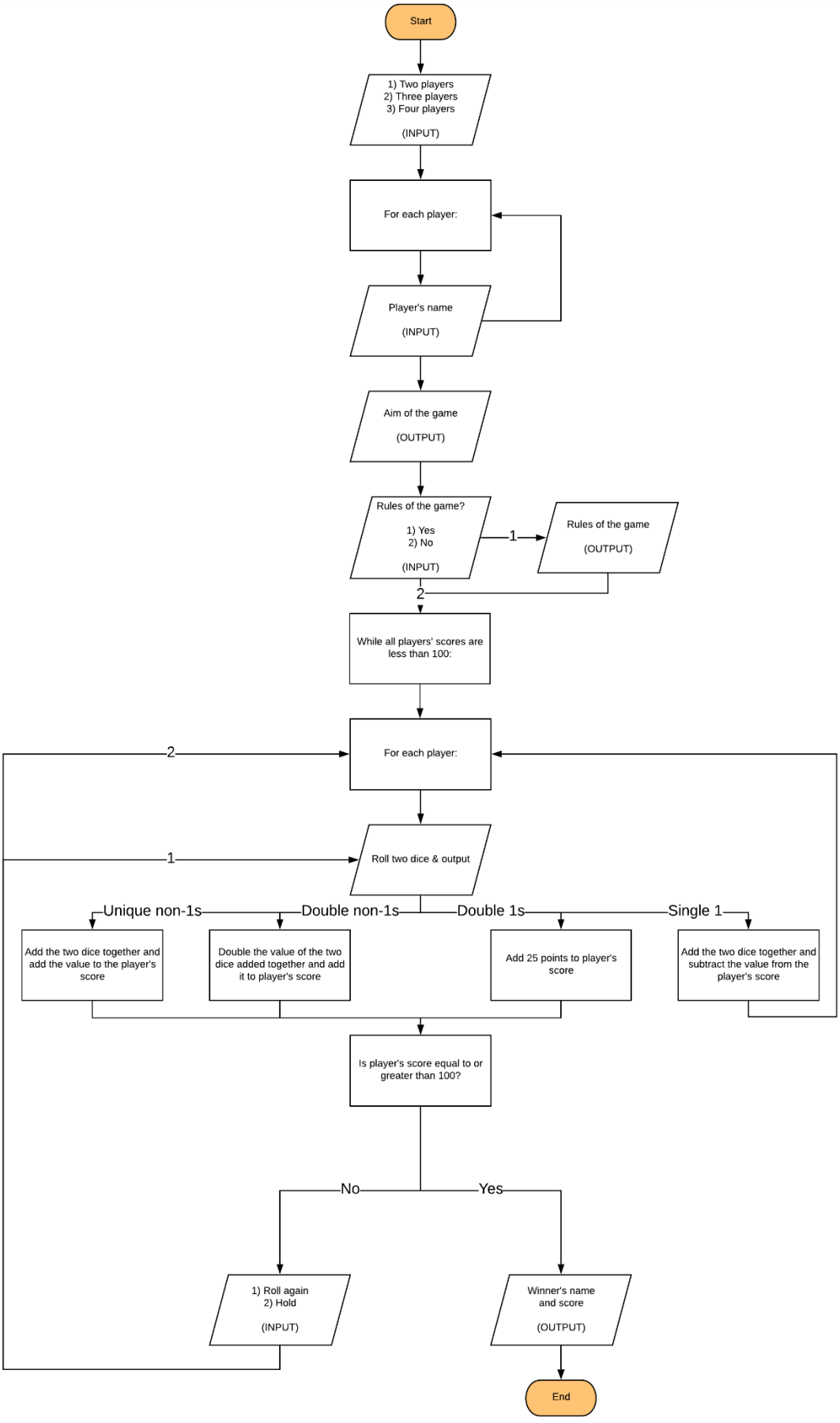
Target Language

This program will be created using Python's IDLE, written in Python 3.7.3.

Basic Process

- Identify the number of players
- For each player, identify names
- Output the aim of the game
- Output the rules of the game
- Player[]'s turn
    - Dice is rolled
    - Execute appropriate action with their roll results
    - Output player's score
    - If player's score is lower than 100, and a single 1 was not rolled:
        - Identify whether to 'hold' or roll again
- Iterate above until score reaches 100
- Output winner's name and score

Flow Chart


Start
1) Two players
2) Three players
3) Four players
(INPUT)
For each player:
Player's name
(INPUT)
Aim of the game
(OUTPUT)
Rules of the game?
1) Yes
2) No
(INPUT)
1
Rules of the game
(OUTPUT)
2
While all players' scores are less than 100:
For each player:
2
1
Roll two dice & output
Unique non-1s
Double non-1s
Double 1s
Single 1
Add the two dice together and add the value to the player's score
Double the value of the two dice added together and add it to player's score
Add 25 points to player's score
Add the two dice together and subtract the value from the player's score
Is player's score equal to or greater than 100?
No
Yes
1) Roll again
2) Hold
(INPUT)
Winner's name and score
(OUTPUT)
End

Data Dictionary

Data item	Data type	Scope	Data location	Variable name	Initial value
Function that outputs the rules of the game	Function	Global	Top level	game_rules	
Function that 'rolls the dice', by generating two random integers				dice_roller	
First rolled die	Integer	Local	dice_roller	rollOne	Random integer
Second rolled die				rollTwo	
Function for user's integer inputs, with validity checking	Function	Global	Top level	user_input	
The maximum value allowed for integer input	Parameter	Local	user_input	rangeMaximum	
The word(s) specific to the purpose/nature of the input				promptWord	
The user's integer input	Integer			inputValue	""
Function for outputting a data structure as a numbered list	Function	Global	Top level	numbered_output	
Data structure which is being outputted	Parameter	Local	numbered_output	dataStructure	

Data item	Data type	Scope	Data location	Variable name	Initial value
Number that corresponds to the option in the data structure, which user inputs	Parameter	Local	numbered_output	number	
Option in the data structure				item	
Function that checks whether a player has won (i.e. score is above specified winning score)	Function	Global	Top level	dict_check	
Dictionary that is currently being checked	Parameter	Local	dict_check	dictName	
Name of the player currently being checked				name	
Score of the player currently being checked				score	
Integer to start counting from in a range	Integer **Constant**	Global	Top level	RANGE_START	1
Compensation for the exclusive upper boundary within a range				RANGE_OFFSET	
Minimum value on the die				DIE_MINIMUM	
Maximum value on the die				DIE_MAXIMUM	6
Minimum value allowed for integer inputs				MINIMUM_INPUT	1

Data item	Data type	Scope	Data location	Variable name	Initial value
Number to start enumerating from when outputting a numbered list	Integer **Constant**	Global	Top level	ENUMERATE_START	1
Compensation for the mis-matched numbering with the list for number of players (i.e. ‘1’ is entered for two players)				NUMBER_PLAYERS_OFFSET	
Compensation for the offset between the index start point and enumeration start point				INDEX_OFFSET	
Point at which the score is to start				SCORE_START	0
Score to reach in order to win				WINNING_SCORE	100
The special number which triggers a different action when rolled (‘1’ as set out in the brief)				SPECIAL_NUMBER	1
The prize which is awarded when both rolled numbers are the special number				DOUBLE_SPECIAL	25
The multiplier applied to the sum of the two rolled numbers when double non-specials are rolled				DOUBLE_XSPECIAL_MULTIPLIER	2
Dictionary that contains the names of the players and their respective scores	Dictionary			playersData	{}
Available options for the number of players	Tuple **Constant**			PLAYER_OPTIONS	(“Two players”, “Three players”, “Four players”)
The number of players that are playing	Integer **Derived**			numberOfPlayers	

Data item	Data type	Scope	Data location	Variable name	Initial value
The player currently being iterated	Parameter	Global	Top level	player	
The inputted name of the player	String			inputName	NONE
Options for whether the rules should be outputted	Tuple **Constant**			RULES_OPTIONS	(“Yes”, “No”)
User’s selection for whether to print the rules or not	Integer			printRules	
Options for whether the player wishes to roll again or hold	Tuple **Constant**			HOLD_OPTIONS	(“Roll again”, “Hold”)
Name of the player currently being iterated	Parameter			name	
Score of the player currently being iterated				score	
Whether to roll the dice or to move on to the next player	Integer			rollOrHold	Index of ‘Roll again’ within HOLD_OPTIONS + INDEX_OFFSET

Pseudocode

START

DEFINE game_rules AS:

this function outputs the rules of the game to the players

PRINT("[NEW-LINE]The rules of the game are as follows:

GAMEPLAY

On a turn, a player rolls the dice repeatedly until either:

- a single [SPECIAL_NUMBER] is rolled; or
- the player chooses to hold (stop rolling)

If either of the above occurs, it is the next player's turn.

SCORING

If a player rolls:

- a single [SPECIAL_NUMBER] – their turn ends, and the total value of both dice is deducted from their score

  [NOTE: It is possible for players to have a negative score eg "-6"]
- double [SPECIAL_NUMBER]s – player gets [DOUBLE_SPECIAL] points
- other doubles – player gets [DOUBLE_XSPECIAL_MULTILPLIER] times the normal amount of points
- in all other cases – the player gets the total of both dice

GAME END

When a player reaches a total of [WINNING_SCORE] or more points (ie the 'winning score'), that player is the winner, and the game ends.")

INPUT("[NEW-LINE]Once you are ready to proceed, press ENTER.")

```
DEFINE dice_roller AS:
        ## this function rolls the dice (generates two random numbers between min & max)
        SET rollOne TO RANDOM INTEGER(DIE_MINIMUM, DIE_MAXIMUM)
        SET rollTwo TO RANDOM INTEGER(DIE_MINIMUM, DIE_MAXIMUM)
        RETURN rollOne, rollTwo

    DEFINE user_input(rangeMaximum, promptWord) AS:
        ## this function is for integer inputs by the user, with validity checking
        SET inputValue TO ""
        WHILE inputValue NOT IN RANGE(MINIMUM_INPUT, rangeMaximum + RANGE_OFFSET) DO:
            TRY TO:
                SET inputValue TO INTEGER INPUT("[NEW-LINE]Enter [promptWord]: ")
                IF inputValue NOT IN RANGE(MINIMUM_INPUT, rangeMaximum +
                RANGE_OFFSET):
                    PRINT("[NEW-LINE][CAPITALISED promptWord] must be between
                    [MINIMUM_INPUT] and [rangeMaximum].")
            EXCEPT:
                PRINT("[NEW-LINE][CAPITALISED promptWord] must be an integer
                between [MINIMUM_INPUT] and [rangeMaximum].")
        RETURN inputValue

    DEFINE numbered_output(dataStructure) AS:
        ## this function is to nicely output a list/tuple as a numbered list
        FOR number, item IN ENUMERATED(dataStructure) DO:
            PRINT("[number]) [item]")

    DEFINE dict_check(dictName) AS:
        ## this function checks if any value within the dictionary is over the winning score
        FOR name, score IN dictName DO:
            IF score IS GREATER THAN OR EQUAL TO WINNING_SCORE DO:
                RETURN TRUE
        RETURN FALSE
```

```
SET RANGE_START TO 1
SET RANGE_OFFSET TO 1
SET DIE_MINIMUM TO 1
SET DIE_MAXIMUM TO 6
SET MINIMUM_INPUT TO 1
SET ENUMERATE_START TO 1
SET NUMBER_PLAYERS_OFFSET TO 1
SET INDEX_OFFSET TO 1
SET SCORE_START TO 0
SET WINNING_SCORE TO 100
SET SPECIAL_NUMBER TO 1
SET DOUBLE_SPECIAL TO 25
SET DOUBLE_XSPECIAL_MULTIPLIER TO 2
SET playersData TO {}

PRINT("Welcome to Double Hog!")
SET PLAYER_OPTIONS TO ("Two players", "Three players", "Four players")
PRINT("[NEW-LINE]How many players are there?[NEW-LINE]")
GO TO numbered_output(PLAYER_OPTIONS)
SET numberOfPlayers TO GO TO user_input(LENGTH(PLAYER_OPTIONS), "your choice") +
NUMBER_PLAYERS_OFFSET

FOR player IN RANGE(RANGE_START, numberOfPlayers + RANGE_OFFSET):
    SET inputName TO NONE
    WHILE inputName IS NOT ALPHA:
        SET inputName TO INPUT("[NEW-LINE]Input Player [player]'s first name: ")
        IF inputName IS ALPHA:
            APPEND playersData WITH inputName : SCORE_START
            BREAK
        ELSE:
            PRINT("[NEW-LINE]Player [player]'s first name must only contain letters
A-Z.")
PRINT("[NEW-LINE]The aim of the game is to be the first player to reach 100 points, and therefore
win the game.")
```

```
SET RULES_OPTIONS TO (“Yes”, “No”)
PRINT(“[NEW-LINE]Would you like to be introduced to the rules of the game?[NEW-LINE]”)
GO TO numbered_output(RULES_OPTIONS)
SET printRules TO GO TO user_input(LENGTH(RULES_OPTIONS), “your selection”)

IF RULES_OPTIONS[printRules – INDEX_OFFSET] IS EQUAL TO “Yes” DO:
    GO TO game_rules

SET HOLD_OPTIONS TO (“Roll again”, “Hold”)
WHILE GO TO dict_check(playersData) NOT EQUAL TO TRUE DO:
## this outer while loop is for when the FOR loop ends, and no one has won yet
    FOR name, score IN playersData DO:
        SET rollOrHold TO INDEX OF “Roll again” IN HOLD_OPTIONS + INDEX_OFFSET
        ## this is to ensure the dice are always rolled at the start of each turn
        WHILE HOLD_OPTIONS[rollOrHold – INDEX_OFFSET] IS EQUAL TO “Roll again” DO:
            SET rollOne, rollTwo TO GO TO dice_roller
            PRINT(“[NEW-LINE]You rolled a [rollOne] and a [rollTwo]!”)
            IF rollOne IS EQUAL TO rollTwo DO:
                IF rollOne AND rollTwo IS EQUAL TO SPECIAL_NUMBER DO:
                    INCREASE playersData[name] BY DOUBLE_SPECIAL
                ELSE DO:
                    INCREASE playersData[name] BY rollOne + rollTwo *
                    DOUBLE_XSPECIAL_MULTIPLIER
            ELSE DO:
                IF rollOne OR rollTwo IS EQUAL TO SPECIAL_NUMBER DO:
                    DECREASE playersData[name] BY rollOne + rollTwo
                    SET rollOrHold TO INDEX OF “Hold” IN HOLD_OPTIONS +
                    INDEX_OFFSET
                ELSE DO:
                    INCREASE playersData[name] BY rollOne + rollTwo
```

IF playersData[name] IS GREATER THAN OR EQUAL TO WINNING_SCORE DO:

PRINT("[NEW-LINE][name] has won with a score of [playersData[name]]!")

QUIT

PRINT("NEW-LINE]You now have a score of [playersData[name]], you need [WINNING_SCORE – playersData[name]] more points to win!")

PRINT("[NEW-LINE]Would you like to roll again or hold?[NEW-LINE]")

GO TO numbered_output(HOLD_OPTIONS)

SET rollOrHold TO GO TO user_input(LENGTH(HOLD_OPTIONS), "your choice")

END

Test Plan

Colour	What is being tested	DIE_MINIMUM	DIE_MAXIMUM	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Is the number generated actually random?	1	6	Expected	Random integer between 1 – 6 (inclusive) e.g. [2, 5, 3, 1, 6]			
			8		Random integer between 1 – 8 (inclusive) e.g. [3, 6, 4, 1, 8]			
			6		Random integer between 1 – 6 (inclusive) e.g. [2, 5, 3, 1, 6]			
			8		Random integer between 1 – 8 (inclusive) e.g. [3, 6, 4, 1, 8]			

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Check if user's integer input is valid and within specified range	LENGTH(PLAYER_OPTIONS) **= 3**	2	Expected	Passes through function, inputValue is returned successfully			
			1	Expected / Boundary				
			3					
			0	Boundary / Invalid	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			4					
			7	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			
			Two	Exceptional				

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Check if user's integer input is valid and within specified range	LENGTH(RULES_OPTIONS) **= 2**	1	Expected / Boundary	Passes through function, inputValue is returned successfully			
			2					
			0	Boundary / Invalid	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			3					
			5	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			
			Two	Exceptional				

Colour	What is being tested	rangeMaximum	inputValue	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Check if user's integer input is valid and within specified range	LENGTH(HOLD_OPTIONS) **= 2**	1	Expected / Boundary	Passes through function, inputValue is returned successfully			
			2					
			0	Boundary / Invalid	Caught by IF statement with error message, WHILE loop triggers and input is re-done			
			3					
			5	Invalid				
			ABCD		Caught by EXCEPT with error message, WHILE loop triggers and input is re-done			
			Two	Exceptional				

Colour	What is being tested	score	WINNING_SCORE	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Checks if the current iteration's score is greater than or equal to the winning score	25	100	Expected	RETURN FALSE			
		99		Boundary				
		100			RETURN TRUE			
		101						

Colour	What is being tested	inputName	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Check if user's input is valid	'James'	Expected	Is accepted by IF statement, is added to the playersData dictionary			
		'John'					
		'James Bao'		Caught by ELSE statement with error message, WHILE loop triggers and input is re-done			
		'1234'					

Colour	What is being tested	rollOne	rollTwo	SPECIAL_NUMBER	Test type	Expected result	Actual result	How it was tested/fixed	Date tested
	Is the scoring system working as intended?	1	1	1	Expected				
		3	3						
		5	5						
		1	5						
		2	6						
		4	3						