FIT3036 COMPUTER SCIENCE PROJECT

Computers Doing IQ Tests, and Pick the 'Odd One Out'

Final Test Report

Tutorial: Assoc. Prof. David L. Dowe

By: Kelvin Benzali

26445468

1 Introduction

The overall approach of testing this project would mainly consist of unit testing, system testing, and user interface testing. The testing stage mainly focused on unit testing due to the importance of each functionality to work in order to solve the problem successfully. The unit testing approach is mainly tested using manual testing environment where the class is called as an object and call the method that want to be tested. The results and method behavior during the testing with various test cases would be monitored and analyzed. The test cases generated for testing the method need to have error and boundary test cases to check how the method handle its limitation or catch any error occurred. However, the testing for odd one out unit has an addition testing approach by using automated testing that can generate questions sample randomly.

The system testing approach is more similar to black box testing where the final program is launched along with its graphic user interface and test the whole functionality of the program at once. In this system testing, the program would be tested for both its functionality to solve the problem, handle inputs and file and the interface response from user action. Since the interface of this program is very simple and basic, the interface testing already included in the system testing as a whole program.

Based on the testing plan proposed in the project specification proposal, the unit testing requirements includes testing the main functionality of each agent which are number, direction, and odd one out agent. The unit testing will be conducted at the early phase of testing stage which started right after the final implementation of the program finished. The user interface testing is also required to ensure the program has a proper graphic user interface and able to interact with the user. The interface testing will be done the same time as the system testing which is at final testing stage of the program. The system testing required to test the program as a whole for its functionality to be able to solve various I.Q. questions problems and to test the odd one out agent knowledge management along with large file input and output functionality. The system testing will be conducted at the last stage of the program.

The requirement for a test to be considered successful is it needs to pass at least 80% of the total test cases proposed to prove that the function will working properly most of the time. The program code coverage cover the most important function such as the agent class solve function. In addition, the question parsing and odd one out learning function that can be considered as the main functionality of the agent. Hence, the trivial function that only support the main function does not need to be tested thoroughly. The criteria that will determine if the program has completed the testing stage successfully if the program has pass all the testing stage includes all the unit testing, system testing and additional basic interface testing.

An additional class that acts as a support for the testing stage is created within the project which are the QuestionGenerator and QuestionGeneratorDriver classes. These classes purpose is to generate a set of random questions specifically for odd one out question to be tested in odd one out agent. The process includes receiving a set of words that would act as the database and generate the questions randomly based on the database then store both the questions and answers into an array. Additional feature is of testing the generated questions into the odd one out agent for both learning and testing is also added into these classes.

2 Test Report

2.1 Unit Testing

Test ID		T01		
Scenario		Checking the question type		
Description		Test the check type function in the agent class which check the question problem type whether it is number, direction or odd one out problem that represented as 1, 2, 3 number respectively.		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	What is the next number of 7 11 13 17 19 23 ?	Return 1	Return 1	Pass
2	Odd one out: Germany, Austria, Australia, Belgium	Return 3	Return 3	Pass
3	walked 15 m towards west. Turned left and walked 20 m. Turned left and walked 15 m. Turned right and walked 12 m	Return 2	Return 2	Pass
4	Testing purposes	Return 0	Return 0	Pass
5	“”	Return 0	Return 0	Pass
Evaluation		The check type function successfully meet the testing criteria by getting pass result on both normal and boundary inputs.		

Test ID		T02		
Scenario		Get the multiple choices		
Description		Test the get multiple choice function in the agent class that has a purpose to get the multiple choice content in the question if any.		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Odd one out: Germany, Austria, Australia, Belgium a.Germany b.Austria c.Australia d.belgium	Fill the multiple choice array with [Germany, Austria, Australia, Belgium]	Multiple choice array becomes [Germany, Austria, Australia, Belgium]	pass
2	What is the next number of 7 11 13 17 19 23 ? A, 27 B, 29 C, 31	Fill the multiple choice array with [27, 29, 31, 33]	Multiple choice array becomes [27, 29, 31, 33]	pass

	D, 33			
3	What is the next number of 7 11 13 17 19 23 ? a) 27 b) 29 c) 31 d) 33	Fill the multiple choice array with [27, 29, 31, 33]	Multiple choice array becomes [27, 29, 31, 33]	pass
4	What is the next number of 7 11 13 17 19 23 ? 1. 27 2. 29 3. 31 4. 33	Multiple choice array becomes empty []	Multiple choice array becomes empty []	pass
5	Odd one out: Germany, Austria, Australia, Belgium	Multiple choice array becomes empty []	Multiple choice array becomes empty []	pass
6	“”	Multiple choice array becomes empty []	Multiple choice array becomes empty []	pass
Evaluation		The get multiple choices function successfully meet the testing criteria of both boundary, error, and normal cases.		

Test ID		T03		
Scenario		Get sequence data in number class		
Description		Test the sequence data function in number class to get the number sequences from the question.		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	What is the next number of 7 11 13 17 19 23 ?	Sequence list becomes [7, 11, 13, 17,19,23]	Sequence list becomes [7, 11, 13, 17,19,23]	Pass
2	What number comes next in this sequence 7, 11, 13, 17, 19, 23?	Sequence list becomes [7, 11, 13, 17,19,23]	Sequence list becomes [7, 11, 13, 17,19,23]	Pass
3	What is the next number of 7	Sequence list becomes []	Sequence list becomes []	Pass
4	What is the next number of 7 and 11 and 13	Sequence list becomes []	Sequence list becomes []	Pass
5	“”	Sequence list becomes []	Sequence list becomes []	pass
Evaluation		The get sequence data function successfully meet the testing criteria for both boundary, constraint and normal cases		

Test ID	T04
Scenario	Get sequence data in direction class
Description	Test the sequence data function in direction class to get both direction and distance data from the question

Test Case	Test Data	Expected Result	Actual Result	Status
1	walked 15 m towards west. Turned left and walked 20 m. Turned left and walked 15 m. Turned right and walked 12 m	Direction list becomes [west, left, left, right] and sequence list becomes [15, 20, 15, 12]	Direction list becomes [west, left, left, right] and sequence list becomes [15, 20, 15, 12]	Pass
2	Turned left for 15 m and walked north for 10 m. Then, turned right for 15 m and continue walked north for 10 m.	Direction list becomes [left, north, right, north] and sequence list becomes [15, 10, 15, 10]	Direction list becomes [left, north, right, north] and sequence list becomes [15, 10, 15, 10]	Pass
3	walked 15 m towards west. Then, turned left, right, and left again with 10, 5, and 20 respectively.	Direction list becomes [west, left, right, left] and sequence list becomes [15, 10, 5, 20]	Direction list becomes [west, left, right, left] and sequence list becomes [15, 10, 5, 20]	Pass
4	Walked for 15, 20, 20, 15, and 10 with directions of north, east, north, west respectively	Direction list becomes [north, east, north, west] and sequence list becomes [15, 20, 20, 15, 10]	Direction list becomes [north, east, north, west] and sequence list becomes [15, 20, 20, 15, 10]	Pass
5	“”	Direction list is empty and sequence list is empty	Direction list is empty and sequence list is empty	Pass
Evaluation		The get sequence data function successfully meet the testing criteria for normal, constraint, and error test cases		

Test ID		T05		
Scenario		Get sequence data in odd one out class		
Description		Test the sequence data in odd one out class to get the odd one out words from the question		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Odd one out: Germany, Austria, Australia, Belgium	Return [Germany, Austria, Australia, Belgium]	Return [Germany, Austria, Australia, Belgium]	pass

2	Odd one out Germany, Austria, Australia, Belgium	Return empty list	Return empty list	Pass
3	Which one is the odd one: Germany, Austria, Australia, Belgium	Return [Germany, Austria, Australia, Belgium]	Return [Germany, Austria, Australia, Belgium]	pass
4	Germany, Austria, Australia, Belgium	Return empty list	Return empty list	pass
5	Odd one out: Germany, Austria, Australia, Belgium, France, Poland	Return [Germany, Austria, Australia, Belgium, france, Poland]	Return [Germany, Austria, Australia, Belgium, france, Poland]	pass
Evaluation		The get sequence successfully meet the testing criteria for all error, normal, and boundary test cases.		

Test ID		T06		
Scenario		Solve the number sequence		
Description		Solve the question problem given the question as an input string and return the solution		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	What number comes next in this sequence 7, 11, 13, 17, 19, 23?	Return 29	Return 29	Pass
2	What number comes next in this sequence 4, 8, 16, 32, 64?	Return 128	Return 128.0	Pass
3	What number comes next in this sequence 8, 27, 64, 125?	Return 216	Return 216	Pass
4	What number comes next in this sequence 111, 99, 105, 93?	Return 99	Return 99	Pass
5	What number comes next in this sequence 500, 175, 7, 512?	Return empty string	Return empty string	Pass
6	“”	Return None	Return None	Pass
Evaluation		The solve function for number sequence able to solve and predict all number sequences that has been implemented in the program successfully. In addition, the program can also handle the boundary and error cases.		

Test ID		T07		
Scenario		Solve the direction question problem		
Description		Solve the question for direction problem given as input text and return the solution for both direction and distance		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	walked 15 m towards west. Turned left and walked 20 m. Turned left and walked 15 m. Turned right and walked 12 m	Return 32 for distance and south for direction	Return 32 for distance and south for direction	Pass
2	walked 15 m towards west. Then, turned left, right, and left again with 10, 5, and 20 respectively.	Return 36.05 for distance and south west for direction	Return 36.0555127 for distance and south west for direction	Pass
3	Walked for 15, 20, 20, 15, and 10 with directions of north, east, north, west respectively	Throw IndexError	Throw IndexError	Pass
4	“”	Return None	Return None	Pass
Evaluation		Direction solve function successfully solve all normal cases. Although the error cases is expected to throw error because of the illogical question, it can be improved by catch and handle the error.		

Test ID		T08		
Scenario		Solve the odd one out problem		
Description		Solve the question for odd one out problem given the question as input text and return the odd one word. However, since the purposes of this testing is unit testing, the correctness of the solution cannot be guaranteed as the machine has not learn any knowledge before hand.		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Odd one out: Germany, Austria, Australia, Belgium	Return either Germany, Austria, Australia, or Belgium	Return Germany	Pass
2	Which one is the odd one out: Germany, Austria, Australia, Belgium	Return either Germany, Austria, Australia, or Belgium	Return Australia	Pass

3	Odd one out: Germany, Austria, Australia, Belgium, France, Poland	Return either Germany, Austria, Australia, Belgium, France, or Poland.	Return Belgium	Pass
4	Germany, Austria, Australia, Belgium	Return empty string	Return empty string	Pass
5	“”	Return empty string	Return empty string	Pass
Evaluation		The solve function for odd one out successfully handled all normal, and error cases.		

Test ID		T09		
Scenario		Odd one out agent learning samples		
Description		Test the learning function of odd one out agent where the agent is given a question and try to learn the question along with a feedback for correct answers.		
Test Case	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Learn Odd one out: Germany, Austria, Australia, Belgium	Australia would hold the lowest relationship points	Australia has -5, Germany has 3, Austria has 3, Belgium has -1	Pass
2	Learn odd one out: Indonesia, Japan, China, Australia and odd one out: Indonesia, South Korea, Japan, China	Australia would hold the lowest relationship point but Indonesia would become the lowest point if Australia is changed to South Korea	Australia has -6, Indonesia has -4	Pass
Evaluation		The learning function successfully make the machine to learn words relationship by using updating the reward into their relationship for each iteration.		

2.2 System Testing

Test ID		T10		
Scenario		System testing of the whole program		
Description		Test the program as a whole including the interface reaction.		
No.	**Test Case**	**Expected Result**	**Actual Result**	**Status**
1	Fill in the text field with "What number comes next in this sequence 7, 11, 13, 17, 19, 23?" and click solve button	Number 29 will show below the answer label as the result	Number 29 show up below the answer label	Pass
2	Fill in the text field with "walked 15 m towards west. Turned left and walked 20 m. Turned left and walked 15 m. Turned right and walked 12 m" and click solve button	Text of 32 south will show below the answer label as the result	Text "32 south" show below the answer label	Pass
3	Fill the text field with "Odd one out: Germany, Austria, Australia, Belgium" and click solve button	Text of either Germany, Australia, Belgium, or Austria will show up below the answer label	Text of "Germany" show up below the answer label	Pass
4	Fill the text with "Testing purposes" and click solve button	No text will show up	No text show up	Pass
5	Click edit knowledge button	Knowledge window show up	Knowledge window show up	Pass
6	Click solve file button	File management window show up	File management window show up	Pass
7	Open the knowledge window and load the "knowledge2000.txt" file in the resources directory, then proceed to open the file window and load "question1.txt" in question slot and "answer1.txt" in answer slot. Then click Run button	The program would load the knowledge successfully and run the file input and show the result of the calculation. The question would show 1000 and the correct would show between 800 and 950.	The program load the knowledge file successfully without error, and load the question file and answer file successfully as well. The result of the calculation show the total question of 1000 and correct answer of 928 , wrong answer of 72	Pass

8	Do the exact action of test case 7 and an addition of loading “output1.txt” file. Click the run button	The result would exactly the same with test case 7 with and addition of “output1.txt” file that would be overwritten with answers list that the program try to answer from “question1.txt”	The result is exactly the same with test case 7 and “output1.txt” is overwritten with the answers list the program tries to solve	Pass
Evaluation		The program has successfully pass the testing criteria for normal cases. The program can handle both normal and error inputs. The program also react accordingly with the user action. In addition, the program also show the capabilities to solve the questions given by the user. The program can handle the file input and output as well.		

3 Conclusion

In conclusion, the program has successfully pass the testing criteria for all unit testing, system testing, and interface testing without a problem. The program is proved to work as intended for all test cases including normal, boundary, and error test cases. Although there are some cases where the program has not yet to implement for handling the error on several areas, the program can still working normally without problem and does not disturb other functionalities in general. Hence, the program can be considered to meet the requirements for the testing stage in general.