Internal Assessment Resource

Digital Technologies Level 3

|  |
| --- |
| This resource supports assessment against:  Achievement Standard 91637v3  Develop a complex computer program for a specified task |
| Resource title: Maths Quiz |
| 6 credits |
| This resource:   * Clarifies the requirements of the Standard * Supports good assessment practice * Follows school’s usual assessment quality assurance process |

Name: Idreis Abdo Due Date: Monday, 9 April

**Candidate’s Statement**

This assessment has been done entirely by me. I have neither copied work from others nor lent my work to others to use for this programme.

Signed Date

Year 13 Assessment

### Achievement Criteria

|  |  |  |
| --- | --- | --- |
| **Achievement** | **Achievement**  **with Merit** | **Achievement**  **with Excellence** |
| Develop a complex computer program for a specified task. | Skilfully develop a complex computer program for a specified task. | Efficiently develop a complex computer program for a specified task. |
| * Designing and implementing a program that includes variables, an indexed data structure, and a modular structure including details of the procedural structures of the modules | * Following a disciplined design and implementation process, with documented cycles of incremental development and… | * Ensuring that the overall modular and procedural design, graphical user interface, and event handling design, are a well-structured, logical decomposition of the task |
| * Testing and debugging the program to ensure it works on a sample of expected input cases. | * …comprehensive testing process to ensure that the program works on inputs that include both expected and boundary cases | * Comprehensively testing and debugging the program in an organised and time effective way to ensure the program is correct on expected, boundary and invalid input cases. |
| * Using class(es) and objects to encapsulate data and methods | As Achieved | As Achieved |
| * Including a working graphical user interface with different sources of event generating components and event handling |
| * Setting out the program code clearly and documenting the program with comments | * Documenting the program with variable and module names and comments that accurately describe code function and behaviour. | As Merit |
| * The programming language for this standard must be a text-based Object-oriented programming language that supports graphical user interfaces (GUIs) and event based programming. | | |
| * The platform for the program may be a personal computer, a mobile device, or a web browser. The program must involve graphical elements such as widgets, images, and/or shapes, and must respond to event-based input from the user. | | |
| * A complex computer program is one that has a modular structure; an indexed data structure (e.g. array or list); input and output; procedural structures that combine sequential, conditional, and iterative structures; a graphical user interface and event handling; and that includes class(es) and objects. Inheritance is not required. | | |
| * A specified task refers to a set task, which requires the development of a complex computer program to resolve. The task must be of sufficient rigour to allow the student to meet the standard and needs to be agreed prior to the program being developed. It may be teacher-given or developed in negotiation with the student. | | |

### Assessment conditions

This assessment will be conducted in *exam conditions*. You will follow a seating plan. Hand in your documentation and programme file at the end of each session with yournameAS91637version number. You will be given **two classes for planning** and **seven classes for programming**. Hand in a copy of your plan before you begin coding. You may use any previous flowcharts and programs you have created.

### Brief

A local educational company wants to give away maths quiz programme to help primary school children. You have been asked to design and implement this programme for 8-12 year old children. The complex programme you are designing must have a graphical user interface (GUI) and use concepts of object-oriented programming.

### Specifications

* The interface should display:
  + Welcome message with brief description of what this programme does
  + User input for name and age
  + Options to choose level of difficulty
  + Set of questions, space to enter answers and gives feedback to user (Timer is optional)
  + User names and scores at the end of each quiz (writing to and reading from text file is optional)
* Error checking and prevention should be implemented as much as possible.
* It is a complex computer program that has a modular structure; an indexed data structure (e.g. array or list); input and output; procedural structures that combine sequential, conditional, and iterative structures; a graphical user interface and event handling; and that includes class(es) and objects.

### Final check

Leave a copy of your planning, daily log and test cases, working programme with comments in S:\DIT\13 DIP Patchigalla\Programming2017\3DIP4. Print page 1 of this assessment task, sign and hand in on due date.

## Appendix 1: Planning Guide

### Task 1: Identify user inputs

*The user can enter their age, name, choose the difficulty through radio buttons, the answers to the math questions. The user can also choose to exit the program if they wish, while also requiring to go next for each question. The button changes from check to next whenever the user needs to click on it, to differentiate from checking your answer to proceeding to the next question.*

### Task 2: Identify information to be displayed

*The 1st frame needs to be displayed showing the title label. It also needs to show the name and age entry fields so the user can write their details, while also showing errors underneath should they enter anything invalid. It will have an exit button and a begin button on the bottom of the frame. The exit button will quit the program entirely while the begin frame will move the frame to the next screen. The error message will display the following: Write a name! (if name field is empty), Letters only! (If any numbers are entered instead of characters only), Invalid number (If no proper number is written or the number field is empty), Ages 8-12 only (If the age is less than 8 or greater than 12 is added).*

*When the begin button is hit the question number will appear across the top of the GUI. The math question will then appear on the left with the check button on the right. The error message will be below the math question. If the user doesn’t enter anything or enters letters the error message will say “Numbers only”. If the user enters the right answer the message displayed will be correct and then the button will change to next. When hitting next a new math question will appear, the new question number will appear, and the button will change back to check. This will continue until there’re 10 questions.*

*Then the summary frame appears. There’ll be a label with a title saying Results, then there’ll be a ScrolledText widget that’ll display all of the past results. It’ll show name, age, difficulty, and results.*

### Task 3: Sketch interface design

*Check sketch diagram.*

### Task 4: Identify any classes required

*The class represents my object orientated program. The modules my class has are the actual mathzquiz itself, where it finds the difficulty selected from the first frame. This then passes into the next module depending on the difficulty easy, medium, hard, and these modules generate a question. Then it passes through the 2nd frame, removing the grid of the first. This module will then have a check button that checks the answer of the question generated and will command grab it from the next module that finds the answer (Ans\_Check()). Depending on this depends on whether the answer written on the Entry in the questions function is correct. If it is correct the next function occurs called the next function. This has a question counter that counts what question you’re up to, while also creating a new question to be answered by going BACK to the function that has the difficulty you chose beforehand. This cycle repeats until 10 questions have been answered. Then the next function comes called results, removing the grid for the 2nd frame and creating a 3rd, this function creates a scrolledtext widget and presents all our information such as age, name, difficulty, and score by calling them from the variable it’s stored. Another module added is the restart module, which is a button added at the summary frame in order to make sure that the code repeats itself consistently. Another function I have is called namecontrol, and it controls what sort of input is allowed for name. This is so we can have error control and ONLY allow text to go through. I have a variable.isalpha() which allows only string to be recognised in the variable, allowing the code to continue to the agecontrol , which is another error control, only allowing numbers to be inserted. If the number inserted is less than 8 or greater than 12, it’ll say that the age isn’t suitable for the program. If no number is inserted or if text is added it’ll say to add a valid number. See UML diagram*

|  |
| --- |
| **class mathquiz** |
| Student : StringVar()  Age : IntVar()  beginbutton: Button  namelabel: Label  nameentry: Entry  agelabel: Label  ageentry: Entry  exitbutton: Button  Radiobutton: Radiobutton  +3 radiobuttons  Welcomelabel: Label  Beginbutton: Button  Instruction: Label  Problem\_label: Label  Answer\_entry: Entry  Check\_ans: button  Next\_btn: button  Feedback: label  Results\_title: label  Scrolled\_display: scrolledtext |
| \_\_innit\_\_(parent:Tk): None  MathQuiz(self): None  Next(self): None  EasyQuestion(self): None  MediumQuestion(self): None  HardQuestion(self): None  Ans\_Check(self): None  Questions(self): None  Quit(self): None  Results(self): None |

### Task 5: Identify any constants or existing data if required

self.difficulty = ["Easy", "Medium", "Hard"] is a constant because my difficulty cannot change within the program

x = random.randint(1,5) random range for numbers in easy difficulty will be the same since the range never changes

y = random.randint(1,5)

x = random.randint(1,12) random range for numbers in medium difficulty

y = random.randint(1,12)

x = random.randint(1,13) random range for numbers in hard difficulty

y = random.randint(1,13)

z = random.randint(1,10)

self.Answer = x+y answer doesn’t change in easy, medium, or hard, same concept as the following

problem = str(x) + " + " + str(y) + " =" problem for easy will always be in this format

problem = str(x) + " x " + str(y) + " =" problem for medium

problem = str(z) + " x " + str(x) + " + " + str(y) + " =" problem for hard

elif age\_error > 12 or age\_error < 8: constant for allowing the ages since the age requirement never changes

### Task 6: Identify indexed data structures

self.difficulty = ["Easy", "Medium", "Hard"]

this is the only indexed data structure we have that contains all the difficulties my program has.

### Task 7: Determine what calculations are necessary

### *def EasyQuestion(self):*

### *x = random.randint(1,5)*

### *y = random.randint(1,5)*

### *self.Answer = x+y*

### *problem = str(x) + " + " + str(y) + " ="*

### *def MediumQuestion(self):*

### *x = random.randint(1,12)*

### *y = random.randint(1,12)*

### *self.Answer = x\*y*

### *problem = str(x) + " x " + str(y) + " ="*

### *def HardQuestion(self):*

### *x = random.randint(1,13)*

### *y = random.randint(1,13)*

### *z = random.randint(1,10)*

### *self.Answer = z\*x+y*

### *problem = str(z) + " x " + str(x) + " + " + str(y) + " ="*

### Task 8: Develop a modular structure for your program

*Def \_\_innit\_\_ initializes my code so that it begins my code. This function has the first frame and all the Entry fields necessary to start the program, i.e. name, age, and difficulty. Then error checking occurs where the function self.namecontrol takes place, checking to see that the entry on my name is all string and no numbers. If this succeeds it moves onto my self.agecontrol function where we check to see if the number added is between 8 and 12, as that is the age allowed for this program. Once this is successful we move onto the self.questions function, where the questions will be displayed. Once the check button is pressed the self.Ans\_Check command is used, where the answer is checked in the function. If the answer is correct green text is shown and it says correct, if the answer is incorrect, red text is shown and it says wrong. Then the next button becomes visible and the command self.Next occurs, where this function allows a next set of random questions from the same difficulty chosen to be displayed. This continues until 10 questions are answered, and then the results function appears. This removes the 2nd frame and makes the 3rd frame appear, getting all the names, scores, ages, and difficulty chosen displayed onto the screen. Then there’s the retry button that once pressed, takes us to the restart function, where the 3rd frame is removed and most variables set are deleted and equalled back to zero, then running back to the first frame, restarting the whole code.*

### Task 9: Define the methods identified

*Refer to my folder please*

### Task 10: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

### Task 11: Refine the plan

*Note any modifications here when iterating through the development cycles.*

*Below is my UML diagram refined to have my error control apart of it this time, as last time it didn’t, and to also include my restart button*

|  |
| --- |
| **class mathquiz** |
| Student : StringVar()  Age : IntVar()  beginbutton: Button  namelabel: Label  nameentry: Entry  agelabel: Label  ageentry: Entry  exitbutton: Button  Radiobutton: Radiobutton  +3 radiobuttons  Welcomelabel: Label  Beginbutton: Button  Instruction: Label  Problem\_label: Label  Answer\_entry: Entry  Check\_ans: button  Next\_btn: button  Feedback: label  Results\_title: label  Scrolled\_display: scrolledtext  Errorcontrol\_label: Label  Retry\_btn: Button |
| \_\_innit\_\_(parent:Tk): None  MathQuiz(self): None  Next(self): None  EasyQuestion(self): None  MediumQuestion(self): None  HardQuestion(self): None  Namecontrol(self): None  Agecontrol(self): None  Restart(self): None  Ans\_Check(self): None  Questions(self): None  Quit(self): None  Results(self): None |

## Another change I made was towards my sketch diagram. Please see Sketch Diagram V2

## I also changed my flowchart, please refer to my folder.Appendix 2: Daily Log and Test Cases

You will need to maintain a **daily log** that documents the development of your programme and also show your testing results. Add a screenshot of your code, any error messages and interface. Describe what was tested, the result, how you fixed the errors.

My daily log:

**27/03/17:** I have identified user inputs, display information and sketched interface design using Visio. I am nearly done with my sketched interface and will definitely be finished by tomorrow.

**28/03/17:** Finished my UML and programming, started programming today.

**30/03/17:** I have added my buttons, first frame, and all the colours

**31/03/17:** I have finished the second frame and added all the buttons to the first. I have to add the mathematical equations to fit my code now.

**3/04/17:** Finished the mathematical equations and made it so that it displays that my answer is right or wrong. Added check and next buttons.

**4/04/17:** Added error control to my name label and age label, and also to my answers, meaning nothing invalid can be put down.

**5/04/17 – 7/04/17:**  At the geography trip so not enough time but I have completed my 3rd frame and added a summary. Still need to make it function.

**8/04/17:** Cannot make my difficulty work for my summary, but everything else is functioning for my summary.

**9/04/17:** Plan is refined and my difficulty function is fixed.

**Test cases:**

|  |  |
| --- | --- |
| **Test cases** | |
| **Input**  **Expected, Boundary, Invalid** | ***Expected output*** |
| Age\_entry (same applies to name\_entry)  -Expected  I expect the user to input an age between 8 and 12, (real text for name\_entry), allowing the program to ask the user the questions.  - Boundary  If the user inputs an input to the age\_entry that is on the boundary of the ages 8-12, I expect the program to function normally.  -Invalid  If the user inputs an input that is an invalid input such as an age that is <8 or >12 or letters instead of numbers such as “five”, I expect the program to display a label to the user, showing an error code of what went wrong. | -Expected      -Boundary  Same as the output above (expected)  -Invalid |
| Radiobuttons  -expected  I expect the user to select one of the radiobuttons that allow you to select either “Easy”, “Medium” or “Hard”, allowing the user to progress to the next screen to solve the equations  -invalid  If the user does not select a difficulty using the radio buttons, I expect my program to display a label telling the user to click and select a difficulty. | -expected      Invalid – as error control I’ve made it so that one button is already  preselected, hence meaning that this is impossible to not select. |
| Ans\_check  -expected  I expect the user to insert some number into the space provided, whether it be incorrect or correct, and my program stating whether it is right or not, and allowing the user to move onto the next question.  -invalid  If the user inputs no answer to the question or only uses letters, the program will tell the user to use numbers only. | -expected    -invalid |

***Appendix 3: Testing log***

|  |  |  |  |
| --- | --- | --- | --- |
| What was test? | How did it go? | | Screen shot |
| Testing to see if my button worked | My radio buttons would not select no matter how many times I would click. Fixed by changing my radio buttons to a forloop, also for more efficiency. |  | |
| Testing to see if forloop for radiobutton worked | It worked, my buttons could now be clicked. | |  |
| Testing to see if hitting the exit button would exit my GUI | Yes, my GUI would exit once I click the button. | |  |
| Testing to see if clicking the begin button would make the GUI exit the 1st frame and move to the second | Yes it worked, but my labels aren’t looking the way I want, especially since I used mathquiz from classwork as a placeholder. | |  |
| Checking to see whether or not math questions would be generated, and seeing if they would be correct or not. | Yes it worked, but my buttons change sizes. Should be related to my modules which is easily fixed. | |  |
| Checking to see if writing invalid name works for error control | Yes it worked. | |  |
| Checking to see if writing numbers in the name section is considered invalid | Yes an error message says that it needs to only be in letters. | |  |
| Checking to see if writing letters in age makes my error control work. | Yes this works as intended. | |  |
| Checking to see if my hard questions are working as intended. | Yes they are working fine, maybe a bit too hard for juniors though. | |  |
| Checking to see if my medium questions generated would work | Yes it is working as intended, meaning that the function is fine. | |  |
| Checking to see if my easy questions generated would work | Yes it is working as intended. | | 5 |
| Testing to see whether or not my 3rd frame would appear after my 10th question | Yes it appears, although this is just a sample screen for it. | |  |
| Testing to see whether my question counter works | Yes it works. My questions go up by 1 everytime I enter a number. | |  |
| Testing to see if my final summary works | Yes, but my difficulty is not displaying. This is a problem with the variable not being displayed properly. | |  |