[](http://www.google.co.uk/url?sa=i&rct=j&q=pates+grammar&source=images&cd=&cad=rja&uact=8&ved=&url=http://www.schooltogs.com/schooltogs&ei=JaI3VYXJBYrhaIyNgYAC&psig=AFQjCNFVEgg6wkOcJvB9BV_eWuOJO_YOEg&ust=1429795749580256)



**PATE’S GRAMMAR SCHOOL**

**COMPUTING DEPARTMENT**

**Unit 3/4 – Programming Project**

CANDIDATE NAME

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

EXAM NUMBER

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

[(1) Anlysis of the problem (10 Marks) 3](#_Toc454442805)

[(i) problem definition 3](#_Toc454442806)

[(ii) Stakeholders 5](#_Toc454442807)

[(iii) RESEARCH THE PROBLEM 6](#_Toc454442808)

[(iv) SPECIFIY THE PROPOSED SOLUTION 9](#_Toc454442809)

[(2) Design OF THE SOLUTION [15 marks] 11](#_Toc454442810)

[(i) DECOMPOSE THE PROBLEM 11](#_Toc454442811)

[(ii) DESCRIBE THE SOLUTION 12](#_Toc454442812)

[(iii) DESCRIBE THE APPROACH TO TESTING 13](#_Toc454442813)

[(3) DEVELOPING THE SOLUTION (25 mARKS) 14](#_Toc454442814)

[(i) INTERATIVE DEVELOPMENT PROCESS 14](#_Toc454442815)

[(ii) TESTING TO INFORM DEVELOPMENT 15](#_Toc454442816)

[(4) EVALUATION (20 mARKS) 16](#_Toc454442817)

[(I) TESTING TO INFORM EVALUATION 16](#_Toc454442818)

[(II) SUCCESS OF THE SOLUTION 17](#_Toc454442819)

[(III) DESCRIBE THE FINAL PRODUCT 18](#_Toc454442820)

[(IV) MAINTENANCE AND DEVELOPMENT 19](#_Toc454442821)

[APPENDIX A - BIBLIOGRAPHY 20](#_Toc454442822)

# (1) Anlysis of the problem (10 Marks)

## (i) problem definition

*(a) Describe and justify the features that make the problem solvable by computational methods.*

*(b) Explain why the problem is amenable to a computational approach.*

Describe your solution. What is it? Why do they need a solution? Why is better than the solution that they currently have or why am I going to emulate a current solution for my clients?

In this section you must talk about why your solution is suited to a computer. You CAN’t just say “well it’s a website, so it can’t be done on paper!”. You must compare to non-computer methods i.e. creating a website for Donuts-r-us means that they will be able to store client details, target advertising, have a much wider audience than simply creating a poster.

You could also say things that a computer is really good at, such as providing multimedia, storing records, analysis of orders & invoices, tracking systems across time, allowing levels of difficulty. You could mention things such as your solution can be developed in a modular manner – i.e. customers, orders, products or sprites, each object (in the game), levels, introduction, help screens etc.

You can also mention things like repeated code – such as several modules being similar (add customer, add employee or baddie number 1, baddie number 2 sharing similar properties). You could mention the use of libraries and that you won’t have to code everything yourself from scratch.

**Example of the solution description**

*“The shoot 'em up is a subgenre of the shooter genre of video games. In a shoot 'em up, the player character engages in a lone assault, often in a spacecraft or aircraft, shooting large numbers of enemies while dodging their attacks. The genre's origins can be traced back to Spacewar!, one of the earliest computer games, developed in 1962 and eventually released in amusement arcades in the early 1970s. However, Tomohiro Nishikado, creator of Space Invaders, is generally credited with inventing the genre.*

*These games are usually viewed from a top-down or side-view perspective, and players must use ranged weapons to take action at a distance. The player's avatar is typically a vehicle under constant attack. Thus, the player's goal is to shoot as quickly as possible anything that moves or threatens him. In some games, the player's character can withstand some damage; in others, a single hit will result in his destruction. The main skills required in shoot 'em ups are fast reactions and memorising enemy attack patterns. Some games feature overwhelming numbers of enemy projectiles and the player has to memorise their patterns to survive. These games belong to one of the fastest-paced video game genres. Large numbers of enemy characters are typically featured. These enemies may behave in a certain way dependent on their type, or attack in formations that the player can learn to predict. The basic gameplay tends to be straightforward and many games offset this with boss battles and a variety of weapons. Shoot 'em ups rarely have realistic physics. Characters can instantly change direction with no inertia, and projectiles move in a straight line at constant speeds. The player's character may collect "power-ups" which may afford the character greater protection, an "extra life", or upgraded weaponry. Different weapons are often suited to different enemies, but these games seldom keep track of ammunition. As such, players tend to fire indiscriminately, and their weapons only damage legitimate targets.*

*Typical games in this genre include: Space invaders, Project starfighter, Galaxian, Defender and Resogun.”*

## (ii) Stakeholders

*(a) Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user).*

Who is it for? What organisation/genre of person. Give some detail e.g. “I am designing a game to appeal to 14-16 year olds who are interested in sci-fi” or “My organization is a restaurant in the Cotswold village of Moreton-in-Marsh, they serve 14 customer per day and have 3 main meals…”.

It’s far easier if you can name a particular person who will providing feedback as you develop your solution. This means that they can comment on your work – ALL the way through your project. So pick someone that you can nag!

In this section, if you have an organisation you need to talk about what they currently do, why it isn’t working and how they intend to use your system in the future. How will it help them? Will it save time, combat existing inefficiencies? Allow them to track data? Stop redundant data? Allow multiple access to data? Allow backups etc. You must be detailed, don’t just say that it will improve things, tell me what things. You could also include things such as cooperate logos, colours etc

If you have a game for a particular audience, make sure that you have thought about what you can and can’t include in your game. If your audience is young, you will have to think about colour schemes, language, playability the kind of things that they can and cannot do. If your audience is older you also need to include things like why they would like the game, have they always wanted a game that….? Are they the greatest fan of …..? You may want to cross reference your work with the PEGI rating system. You may want to mention that you will have to design the levels to increase in difficulty or even have a tutorial level that presents instructions to users (pants for under 8s, but useful for over 16s).

You could also talk about the kind of platform that you are writing for. Does your organization have three laptops? Does it have two desktops? Are you designing for a phone, tablet? Are there resolutions that you can or can’t use?

## (iii) RESEARCH THE PROBLEM

*(a) Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.*

*(b) Describe the essential features of a computational solution explaining these choices.*

*(c) Explain the limitations of the proposed solution.*

When you conduct your research you will have to add detail. For creating a game, this is relatively simple. Think about colours, levels, playability, what did you like, what did you hate, what can you include, what is just beyond your capability, what extensions could you add. You must describe the games that you test and then evaluate each one – I would suggest that three would be sufficient. Go into detail - What are the controls? Who are the enemies? What is the aim? What is the jeopardy? How do you progress? Are there lives? Is there a score? What are the graphics? What platforms is it available for? Are there sounds? DO you have weapons & how are they used?

If you are looking at industrial software, this can be harder. If you can get hold of demo software, then use this and see how it works. You may have to think slightly laterally. For example, if you are developing a booking system for theatre tickets, you may have to look at similar industries, such as cinemas, events & travel. You may have to rely in the blurb that they have on the website – with video adverts etc. Include screenshots so that your examiner gets an idea of the software that you have been researching. But your analysis will be similar to those looking at games. What features does it have, which can you do, which can’t you do, which do you like and those that you don’t.

You must also mention what you cannot do – creating a limitations heading would be good. What will you be unable to do & why. E.g. I will develop my system for a Windows environment, because I only have access to Windows based computers. I will have to limit my levels to 7, because I won’t have time for 300. I will have to encrypt my data using a simple Ceaser Cypher, because it will take too long to write an encryption algorithm.

You must also include some information about the computational solution. Here is an example:

### *Abstraction and visualisation*

* *What are the key objects in your game?*
* *What sprites, symbols, icons or font sets will be needed?*
* *Are there further lines of investigation to follow?*

### *Thinking ahead*

* *•What will your inputs be? e.g. cursor keys, WSAD, mouse etc.*
* *•What will your output screens be? e.g. main menu, story, controls, level select, game screen, game over screen, high score.*

### *Thinking procedurally*

* *•Identify the 'game states' you will need. e.g. menu, load level, playing, game over.*
* *•Break the problem down into what will happen in each of the game states for OnCreate(), OnUpdate(), and OnRender().*
* *•Pipelining: Decide the order of each component of the game. e.g. in 'playing' game-state: get inputs, move object, collision detection, movement correction. In OnRender() draw player, draw missiles, draw enemies.*
* *•Explain that the program loops continually around OnUpdate and OnRender.*
* *•State that these sub-problems make the project amenable to a programmed solution.*
* *•Will you have any reusable program components? Functions used for more than one purpose.*

### *Thinking logically*

* *•What are the critical if statements?*
* *•What are the critical loops?*

### *Thinking concurrently*

* *•Are there aspects of the game where more than one thing happens at once? The classic example is the sound player using multi-channel sound to play different sound effects and background music at the same time as updating the game logic.*

## (iv) SPECIFIY THE PROPOSED SOLUTION

*(a) Specify and justify the solution requirements including hardware and software configuration (if appropriate).*

*(b) Identify and justify measurable success criteria for the proposed solution.*

You must have detail here, because this will form a substantial part of your evaluation and is also your success criteria. If you just say “I will create a game”, then your evaluation will be “I created a game”!. You will also need to list the hardware and software requirements. This will include stations (laptops/PC.Tablets etc), the OS, any software needed (e.g. Python), any peripherals (printers etc) and networking.

**Here is an example of solution requirements:**

Following this feedback, I have made some changes to the design objectives. The new objectives can be seen below. The altered parts have been highlighted.

1. Aesthetic objectives
   1. The window size is 800x600 pixels and the window should be non-resizable
   2. The colour scheme is a light blue background with white and yellow text
   3. The monster character will be a dark blue/purple colour with a hair-like texture
   4. Text boxes where text must be entered will be filled white rectangles
   5. Text in text boxes will be black
   6. The font used will be Arial Rounded MT Bold or similar and will be consistent throughout the software
   7. Action buttons
      1. These have yellow text in all capitals
      2. They have square corners
      3. They will have a slight raised effect
   8. Quick Fire Questions
      1. The questions appear one at a time
      2. The questions are white and in large writing
      3. The answer is entered in a text box at the bottom of the screen
      4. Right answer – ‘Correct!’ is displayed on the screen in green writing
      5. Wrong answer – ‘Incorrect!’ is displayed on the screen in red writing
2. Inputs
   1. Screens are navigated and actions are carried out by clicking on-screen buttons
   2. There are text entry fields on the following screens:
      1. Student Log In
         1. Information to be entered - username
      2. Teacher Log In
         1. Information to be entered - password
      3. Add Student
         1. Information to be entered – first name and last name
      4. Delete Student
         1. Information to be entered – student username
      5. View Students
         1. Information to be entered – student username
      6. Game Screen
         1. Information to be entered – answer
   3. Check boxes are used to select times tables
3. Processing
   1. The software will need to change the questions used in the game depending on the times tables selected
   2. Quick Fire Questions
      1. Random questions will be generated
      2. The software needs to determine whether the given answer is correct
      3. The software needs to record the number of correct answers given
      4. A timer is used to time how long it takes for 20 correct answers to be given
   3. The software will need to update the high scores if necessary at the end of the game
4. Outputs
   1. Quick Fire Questions
      1. The question is displayed
      2. The number of correct answers out of 20 is displayed
      3. The timer is displayed
      4. Whether an answer was right or wrong is displayed
      5. The final score is shown at the end of the game
   2. High Scores
      1. Displays top 5 times for the game
   3. View Students
      1. Displays list of all student usernames on screen
      2. Must be able to display at least 45 students
      3. If a username is entered, a pop up appears with that user’s details
   4. Pop ups are used to alert the user if a wrong username has been entered, or if a field has been left empty
      1. These should have a design consistent to the main program window

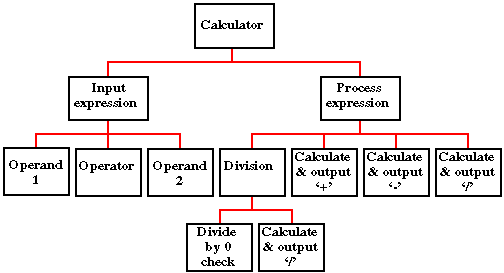
# (2) Design OF THE SOLUTION [15 marks]

## (i) DECOMPOSE THE PROBLEM

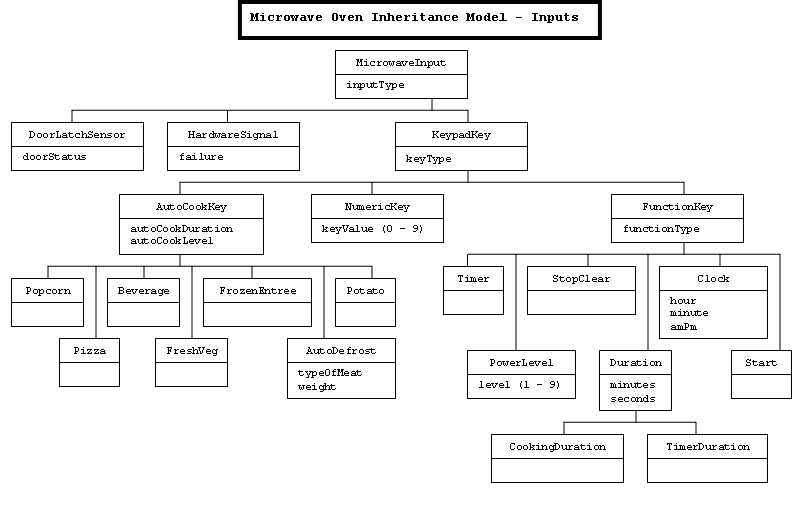
*(a) Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.*

Choose and justify your choice of language – it is NOT good enough to say I will use Python cause school have it and I can use it. That can be mentioned, but you should say – I will use python because it has libraries that I can utilise, such as….I am not too concerned about the GUI element of my solution…..The file handling in Python is relatively easy…..I can adapt Python to become an OOP and this will help me because – this class will contain this object and this object and this object, reducing the amount of code that I need. I will use python in a procedural manner because my project will achieve this task and then this task and so on which lends itself to the procedural nature of this language.

Include a top-down diagram (or inheritance/class diagrams for OOPS) to describe your project and how it will interact. You could include other DFDs – including Entity diagrams Activity diagram, use case. You will include flow diagrams for your work. But I would recommend in

e.g. [](https://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj7kZjK5aDQAhWFvhQKHdKaAU0QjRwIBw&url=https://cgi.csc.liv.ac.uk/~frans/OldLectures/2CS21_Ada/week6/case.html&psig=AFQjCNGrT6KY2oFg0osGeCvVfvPJ3DUbAw&ust=1478956761578040)

Top down diagrams are used for procedural programming languages.

[](http://www.google.co.uk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj36vnN46DQAhVHuxQKHYFiDnsQjRwIBw&url=http://sce.uhcl.edu/whiteta/sdp/createSystemModel.html&bvm=bv.138169073,d.d24&psig=AFQjCNG_CwXn9ZZpnlO4kOYDLwDjGbHzUg&ust=1478956117139706)

Class/Inheritance diagrams are for object orientated code, you do not need to do this if you are using a procedural language

Try and reduce this diagram to it’s smallest component parts.

You may wish to reference some of the computational think in this section by refereeing to:

* Abstraction – what have you discarded?
* Thinking ahead – what problems/limitations can you foresee?
* Thinking concurrently – quickly dismiss any thought of using cores independently
* Reference the next section to thinking procedurally

## (ii) DESCRIBE THE SOLUTION

*(a) Explain and justify the structure of the solution.*

*(b) Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.*

*(c) Describe usability features to be included in the solution.*

*(d) Identify key variables / data structures / classes justifying choices and any necessary validation.*

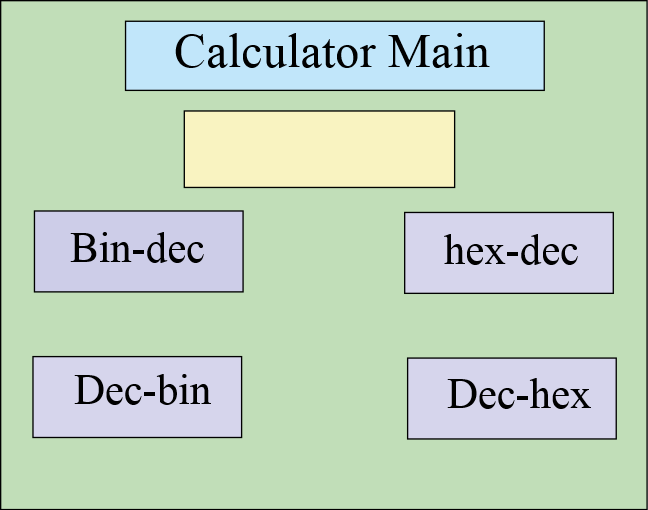
*(e) Identified and justified the test data to be used during the iterative development of the solution.*

Explain any connections, dependencies or links between parts of you design e.g. the main page will have a login drawing on data from users table.

Modularise you work e.g. complete these elements for each of your modules (screens)

Using the example above – the calculator

**Calculator – Main Page**

****

The main page will contain the options for my binary calculator. I will have four options that the user is allowed to choose (bin-den, den-bin, hex-den, den-hex) and they can enter the number onto the front page. It will also store their number in a file for later use. There is a title, data entry for number buttons……

Remember to list all of the variable/functions used, their purpose, data types and validtions

**Functions and variable used:**

Var click\_me Boolean 0 or 1 validation – lookup check 0 or 1

Var sci\_bin String (3) – validation – length check (3 chars)

Func\_conv\_bd(number) function that converts binary numbers to denary numbers

**Flow chart to show the process**

START

Bin to den?

Den to bin?

Remember to include routines for the validations

*……………………….etc*

**Pseudo code**

Proc calculator\_main

Open main\_page

If x < 7

Open random\_message(“this must be a binary number”

End if

………………………………….etc

In order to test the functionality of this page, I will test the following items

**Button1**

Should close the main page and open the denary – binary converter

**Button 2**

Should close the main page and open the binary – denary converter

………………………etc

You need to plan the tests that you will use as you are developing this module – it can be quite short – tests should be designed to demonstrate that your system is working – particularly validations and the logic of the module.

**Test Plan for development**

**For var\_number**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| 010100101 | valid | Saves number |  |  |
| 01010100101001010100 | borderline | Saves number |  |  |
| 000000000000 | boarderline | Saves number |  |  |
| bob | Invalid | Rejects text |  |  |

………..etc

## (iii) DESCRIBE THE APPROACH TO TESTING

1. *Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.*

This is the overall testing for the system and should be a test plan that proves that you have completed the system and met all of your success criteria. You need to describe your methods of testing – these could include alpha, beta, white box, black box, top down & bottom up. There are several ways in this should be achieved:

1. System test – derive three scenarios that need to be completed, e.g. a customer needs their details entered, they wish to make a purchase and get their receipt. Choose one to be valid, the next to be borderline and the last to have invalid data in it.
2. Ask you user to test your system. Give them two tasks to do so that they can experience the usability – acceptance testing (black box)
3. White box testing – stress test variables & functions to try and expose weaknesses, e.g.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test data | Test type | Predicted outcome | Actual outcome | Pass/Fail |
| 010100101 | valid | Saves number |  |  |
| 01010100101001010100 | borderline | Saves number |  |  |
| 000000000000 | borderline | Saves number |  |  |
| bob | Invalid | Rejects text |  |  |

# (3) DEVELOPING THE SOLUTION (25 mARKS)

## (i) INTERATIVE DEVELOPMENT PROCESS

*(a) Provide annotated evidence of each stage of the iterative development process justifying any decision made.*

*(b) Provide annotated evidence of prototype solutions justifying any decision made*.

## (ii) TESTING TO INFORM DEVELOPMENT

*(a) Provide annotated evidence for testing at each stage justifying the reason for the test.*

*(b) Provide annotated evidence of any remedial actions taken justifying the decision made.*

# (4) EVALUATION (20 mARKS)

## (I) TESTING TO INFORM EVALUATION

*(a) Provide annotated evidence of testing the solution of robustness at the end of the development process.*

*(b) Provide annotated evidence of usability testing (user feedback).*

## (II) SUCCESS OF THE SOLUTION

*(a) Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.*

## (III) DESCRIBE THE FINAL PRODUCT

*(a) Provide annotated evidence of the usability features from the design, commenting on their effectiveness.*

## (IV) MAINTENANCE AND DEVELOPMENT

(*a) Discuss the maintainability of the solution.*

*(b) Discuss potential further development of the solution.*

## APPENDIX A - BIBLIOGRAPHY