

**NORTH SYDNEY BOYS HIGH SCHOOL**

**COMPUTING STUDIES DEPARTMENT**

**SOFTWARE DESIGN AND DEVELOPMENT**

HSC ASSESSMENT TASK - PERSONAL SOFTWARE DESIGN PROJECT

(30% of HSC Subject Assessment Mark)

Extract from Syllabus:

*Developing a Solution Package*

*The project(s) in the HSC course is intended to reinforce the content covered in the other topics in the course. Students need to experience working as part of a team, as this is common in the computing field beyond school. In order to be able to develop software successfully, students need to be able communicate well with others and to act in a social and ethical way. The project is one area in which students may be given these opportunities. The project(s) will build students’ understanding of the content dealt with in the other topics in the course and should be undertaken throughout the duration of this course.*

Outcomes

*A student:*

H1.1 explains the interrelationship between hardware and software

H1.2 differentiates between various methods used to construct software solutions

H1.3 describes how the major components of a computer system store and manipulate data

H3.1 identifies and evaluates legal, social and ethical issues in a number of contexts

H3.2 constructs software solutions that address legal, social and ethical issues

H4.1 identifies needs to which software solutions are appropriate

H4.2 applies appropriate development methods to solve software problems

H4.3 applies a modular approach to implement well-structured software solutions and evaluates their effectiveness

H5.1 applies project management techniques to maximise the productivity of the software development

H5.2 creates and justifies the need for the various types of documentation required for a software solution

H5.3 selects and applies appropriate software to facilitate the design and development of software solutions

H6.1 assesses the relationship between the roles of people involved in the software development cycle

H6.2 communicates the processes involved in a software solution to an inexperienced user

H6.3 uses a collaborative approach during the software development cycle

H6.4 develops effective user interfaces, in consultation with appropriate people.

You are required to select a problem/project approved by your teacher which has to be solved using the *structured* software development approach. You are then required to design and build a working solution using a suitable programming language (e.g. Visual Basic) or one approved by your teacher. Your solution must include full documentation of all stages of the Software Development Cycle and satisfy the Board of Studies requirements. See *BOS Support and Specifications Documents*:

Refer:

<http://www.boardofstudies.nsw.edu.au/syllabus_hsc/pdf_doc/softwaredesign_support.pdf>

<http://www.boardofstudies.nsw.edu.au/syllabus_hsc/pdf_doc/softwaredesign_specs.pdf>

**Steps in Project Management**

Throughout the project you are expected to complete the standard software development stages, and submit material periodically which relates to each stage:

(1) Defining and Understanding the Problem

(2) Planning and Design

(3) Implementation

(4) Testing and Evaluation

(5) Maintenance

**Project Ideas**

Stock market simulator Calculator

Student diary Study program system

Appointment system Hotel bookings system

Video store management system Subject choice/report system

Payroll system Cloze test

Stock control Batch banking/ audit system

Text encryption/ analysis Olympic Games ticketing

Library system Connect Four

Mastermind Snakes and Ladders

Hangman Clock Patience

Memory Battleship

Yahtzee Draughts

Search a Word Monopoly

Bingo Ludo

Guess Who Cluedo

Sorry Trouble

Crossword generator Quizzes

Drill and practice games Idiom/Colloquial suggester

Assembler Route Optimizer

Athletics carnival software Resource allocation

Cash register Tenpin bowling scoring

Calculation of sport averages Wedding planner

**Project Assessment**

Due Dates indicate the **final deadline** of Friday 1:35pm in the week stated. You are advised to make an appointment with your teacher before the final date indicated for each Check.

If you miss any of the dates indicated below – you need to contact your teacher immediately on return to school and present the necessary medical certificate. If you are involved in any organised school activity on the dates indicated below – you have to arrange an interview with your teacher before the Friday deadline.

Indicate your **Name** and the **Check Number** clearly on all written work. [Good idea to include in a header/footer.]

|  |  |  |
| --- | --- | --- |
|  | Due Date | Weighting |
| **Walkthrough** | *Term 4 Week 10 2013* | 5% |
|  |  |  |
| **Check 1: Defining and Understanding the Problem** | *Term 1 Week 4 2014* | 5% |
|  |  |  |
| **Check 2: Planning and Design** | *Term 1 Week 7 2014* | 10% |
|  |  |  |
| **Check 3: Implementation** | *Term 2 Week 6 2014* | 10% |
|  |  |  |
| **Check 4: Testing and Evaluation / Maintenance** | *Term 3 Week 1 2014* | 10% |
|  |  |  |
| **Completed Final Project** | *Term 3 Week 4 2014* | 60% |

**DETAILS OF STAGES**

**Walkthrough**

You must arrange an appointment with your teacher and discuss the following aspects of your software assessment proposal:

* Definition of the problem to be solved, needs of the ‘client’; boundaries (scope) and an outline of how you intend to solve the problem – submitted to your teacher in writing
* Brief discussion of the approaches and ideas considered and rejected.

Your teacher will be looking for a balanced feasibility study in your consideration of problem and solution offered.

* Printed Gantt Chart showing dates by which you intend to complete each stage shown in the Project Assessment Check Stages
* Discuss the programming language you intend to use and provide a timeline to your teacher that shows how you intend to gain mastery of the language (if you are not already confident in the language)

This interview will last about 10 minutes

**Checks 1 thru 4**

Documentary evidence is to be presented showing completed work for each stage of the Software Development Cycle. All text documentation is to be word processed, diagrams may be hand-drawn. Include all work completed from prior stages.

Your project logbook will be marked. You are expected to make a minimum of one entry per week. **The logbook must be hand-written.**

**Final Completed Project**

All documentation presented in the earlier stages must be submitted as hard copy in a single folder. Intermediate stages of coding are to be included in your printed documentation.

Your printed material must have a table of contents which lists the page numbers for the different aspects. Include a footer and bibliography.

**A Checklist for the Software Development Cycle**

*Note that some tools may be used in more than one stage of the cycle.*

Check 1 : Defining and Understanding the Problem

* + A problem statement and initial design specification
  + Screen designs
  + Storyboard showing the initial user interface design
  + Context Diagram
  + Data Flow Diagram (initial version)
  + Gantt Chart of entire project
  + Social / Ethical considerations (e.g. gender bias, copyright, ergonomic, accessibility and plagiarism issues)
  + Logbook of tasks completed, problems encountered, etc.

Check 2 : Planning and Design of Software Solutions

* + Discussion of selection of language to be used (as per Walkthrough)
  + List ALL Functions, Procedures and Modules with a brief description
  + Pseudocode and algorithm (system) flowcharts of ONE module
  + IPO Chart of the main module
  + Structure Chart of ALL modules
  + Define the Files that will be used with their respective Data Structures
  + List the key Variables with their data type and a brief description
  + Data Dictionary (initial version), to include Data Types/Classes
  + Platform/OS considerations and hardware implications of your project
  + Provide ONE example each of a Decision Tree and a Decision Table
  + Prototype of system to be available for demonstration. E.g. interface and basic functionality
  + Test data that will enable testing of all paths through the algorithm as well as the testing of each boundary condition. If test data for the entire algorithm is not feasible, select one module and create extensive test data for it.
  + Logbook

Check 3 : Implementation of Software Solutions

* + Justify and discuss choice of user interface items selected – e.g. menu, command buttons, check boxes, etc.
  + EBNF and Railroad diagrams
  + Updates (i.e. additional detail) to DFDs and Data Dictionary
  + Selected test data with the corresponding expected output
  + Error checking: Demonstrate use of stubs, flags, debugging output statements
  + Evidence of testing of all pathways, desk check and peer check
  + Evidence of breakpoint, traces and single line stepping
  + Readability of code
  + Discussion of syntax errors, runtime errors, logic errors
  + Documentation: developer, intrinsic, user, technical, installation, online help
  + Compile to executable application or installer-type application

Check 4 : Testing and Evaluation of the Solution

* + Selected test report with test data: pathways, boundary conditions, syntax, logic and expected output
  + Comparison with original specification
  + Efficiency and elegancy of code
  + Describe how alpha and beta testing might be used
  + Refine the user interface

[Maintenance of Software Solution]

* + Describe suitable CASE tools to monitor changes and version control
  + Explain how you would cater for possible changing user requirements
  + How would you handle updates to the code? E.g. changes to the environment
  + Logbook to record any changes in code, design, etc.

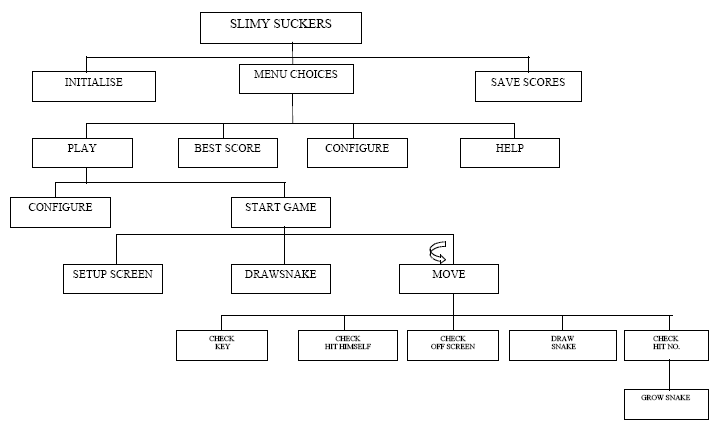
Final Project (written component) :

* + Fully updated Storyboard
  + User Manual
  + Technical / Installation Manual
  + Screen shots
  + Printout of a sample of source code
  + Project report outlining what you have learnt from your work on this project, future directions and modifications of your software
  + Bibliography that details sources of information/ideas, assistance with coding, etc.
  + Peer Check reports
  + Completed Logbook

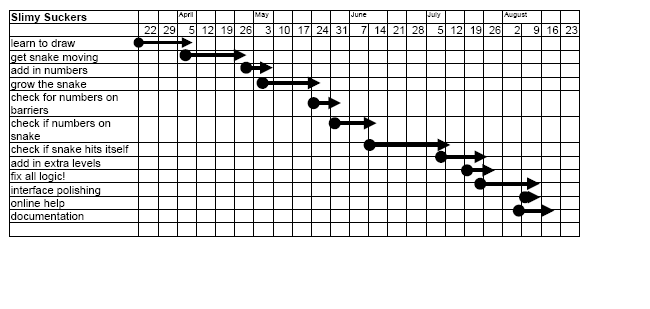
Breakup of Marking Criteria for Final Project

|  |  |  |
| --- | --- | --- |
| Printed documentation - user and technical/installation manuals, screen shots |  | 20% |
| Project report |  | 5% |
| A working tested program with all associated documentation to be submitted on a CD or USB thumb drive. |  | 30% |
| You will be required to present all aspects of your final project to the class. |  | 5% |
| Total |  | 60% |

## Example of a Structure Chart



## Example of a Gantt Chart



## Example of Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **ELEMENT** | **LENGTH** | **TYPE** | **DESCRIPTION** |
| again$ | 1 | Alpha | Asks the player whether he'd like to play again |
| back |  | numeric | variable name of the square dimensioned (back square) |
| backrou$ | 1 | Alpha | colour of the background |
| backrou |  | numeric | colour of the background |
| colourl | 10 | Alpha | colour of the border |
| colour2 | 10 | Alpha | colour of the snake |
| cover |  | numeric | used so the next number doesn't appear where the snake is located |
| falg |  | numeric | a flag set to know whether to increment the variable 'numb' |
| flag |  | numeric | program will drop out of the 'move' loop to perform a different task |
| helpch |  | numeric | choice from help menu |
| increase$ | 1 | Alpha | whether the speed increases through the game |
| index |  | numeric | An index used in "Configuration'9 to be index to direction$(5) |
| keypress |  | numeric | the ACSII value of keypress 1$ |
| keypress 1$ | 1 | Alpha | the key entered by the player to make his snake move |
| lastkeypress |  | numeric | before a new keypress is entered, the old one becomes lastkeypress |
| Lives |  | numeric | How many lives the player has left |
| mmenu |  | numeric | choice from main menu |
| Nam$ | 15 | Alpha | name of player |
| Nexts |  | numeric | The level number that the player is up to |
| Numb |  | numeric | The counter of which is the next number the snake must 'swallow' |
| pause |  | numeric | flag set to say the game was just paused |
| rectg |  | numeric | variable name of the square dimensioned (front square) |
| rect |  | numeric | variable name of the square dimensioned (front square) |
| Rndmx |  | numeric | a random horizontal value where a number will be placed to be collected |
| Rndmy |  | numeric | a random vertical value where a number will be placed to be collected |
| Shape |  | numeric | The choice of shape by the user, It determines the shape of the snake |
| score! |  | numeric | the players score throughout the game |
| snake$ | 20 | Alpha | name of player's snake |
| speed |  | numeric | speed at which the snake moves |
| pole |  | numeric | a flag set to tell if the validation of the keys was OK |
| upto |  | numeric | A count to see what part of the subroutine it is up to |
| vpos |  | numeric | the vertical position of an error message |
| hpos |  | numeric | the horizontal position of an error message |
| x |  | numeric | the current position of the snake (horizontal) |
| Y |  | numeric | the current position of the snake (vertical) |
| **ARRAYS** |  |  |  |
| bestname$ | 10 | array | array where top ten players are sorted and then written to disk |
| bestscores | 10 | array | array where top ten scores are sorted and then written to disk |
| direction | 5 | array | these keys are converted to ASCII values |
| direction$ | 5 | array | player chooses the keys he wishes to play with |
| done | 10 | array | an array pointing out if a message has already been done |
| startx | 640, 200 | array | The starting "x" position of the barrier and the number barrier |
| Starty | 480, 200 | array | The starting "Y" position of the barrier and the number barrier |
| endx | 640, 200 | array | The end "x" position of the barrier and the number barrier |
| endy | 480, 200 | array | The end "y" position of the barrier and the number barrier |
| snake$ | 640, 480 | array | Horizontal and Vertical Position of the snake |
| Lengt | 9 | array | As each number is picked up, the snake grows to this certain length |
| Randm | 10 | array | an array containing 10 different random messages |