|  |
| --- |
| 25th March 2019: Update to Assessment brief in light of COVID-19 situation: |
| We are aware that there are a very small number of students who don’t have access to suitable technology to work on their unit 2 assessment. Please see below for alternatives:   1. **Online Compilers for C++** : For those students with access to a laptop or PC, but incompatible operating system or slow processor, you can write C++ code and compile it online at any of these websites    * <https://onlinegdb.com/S198nLPU8> (you will need to register with this website if you want to save and restore your work)    * <http://cpp.sh/>    * <https://www.jdoodle.com/online-compiler-c++/>   For students using this option, please don’t worry about uploading a .exe with your submission to My Learning Space   1. **Alternative Submission Criteria:** For those students who are unable to get the above to work, we have identified alternative evidence that you can submit for the darts coursework instead of a .exe and source code, described below:    * UML Diagram MUST contain the following for each class:      1. all variable names and variable types      2. function headers      3. Relationships between classes must also be included.    * Pseudocode must be provided for each class (including the class that would contain your int main() function) and this should be very descriptive.    * In place of source code and .exe, you should create a .txt file for each class and header file, and include as much code/description/comments as you can in here. This should include any libraries that you need, variables (including public/private type), object declarations, function prototypes, function headers, and function body. We appreciate that the text may not be syntactically correct, as you won’t have the benefit of a compiler giving you feedback on your solution. However, we will be using this as an indicator of the approach that you’ve taken to solve the problem and how efficient and effective this is.    * Your report and form, as described in the coursework brief should still be submitted    We encourage any students who are adopting this approach to only concentrate on Task 1 due to the complexity of Tasks 2 and 3. |

|  |
| --- |
|  |
|  |
| **School of Design & Informatics** |

**Assessment Instrument Coversheet**

Module Code: CMP102

Module Title: Software Design

Lecturer: Dr Suzy Prior and Dr Natalie Coull

Submission Date: 23rd April 2020 at 11.59pm

Feedback Return Date: Feedback will be provided within 15 working days

Feedback Type: via My Learning Space

(eg verbal, My Learning Space)

Grading Criteria Refer to Page 7

**Submission Requirements:**

Your assessment must be submitted via My Learning Space. The maximum file size which can be submitted is 20MB so you may need to reduce the size of any image files within your document.

Guidance on submitting via My Learning Space is available at: <http://submit.ac.uk/en_gb/training/student-training/submitting-a-paper>, but please contact the Support Enquiry Zone on 01382 308833 or [sez@abertay.ac.uk](mailto:sez@abertay.ac.uk) if you have any problems with submitting your work on the My Learning Space.

Submission of your work after the submission date deadline will be deemed as late submission and will incur penalty, including the possibility of the work being awarded a non-submission (NS) grade.

**CMP102 Software Design**

**1 Introduction**

Module CMP102A Software Design is assessed by a combination of quizzes, practical exercises, and demonstrations of your completed tasks, which show your skills and evidence achievement across the module.

**2 Learning Outcomes**

All modules at Abertay are defined by module descriptors which are published on Oasis. In each case the module seeks to encourage the student to achieve “Learning Outcomes” and this provides the focus, in particular, of assessments – you need to show that you have achieved them. For CMP102A these outcomes are:

1. Communicate the ideas and languages of software design in a professional manner.  
2. Confidently break down design problems using problem-solving techniques.  
3. Develop, understand and communicate designs for software projects using object-oriented design techniques.  
4. Develop, understand and communicate designs for databases using relational database design techniques.  
5. Select appropriate modelling techniques for an application

By the time you have completed the assessed exercises across the term, you will have evidenced all of these outcomes.

**3 Assessment**

Your assessment for this module comprises two units of assessment. Each unit of assessment is worth 50%. You must evidence all of the learning outcomes in order to pass the module, this means you must complete the SQL quizzes and UML exercises in order to pass.

Unit 1 (worth 50% of the module) consists of 6 components. Throughout the semester you will complete exercises and quizzes which contribute to creating Unit 1 (your portfolio). Weekly quizzes (50% of Unit 1)

1. Quizzes
   1. Object Oriented quizzes from weeks 2, 3, 4 and 5 The quizzes in total are worth **20%** of your grade for this unit. You can also find links to the quizzes within the weekly folders.
   2. SQL quizzes from Weeks 8, 9 and 10 The quizzes in total are worth **30%** of your grade for this unit. You can also find links to the quizzes within the weekly folders.
2. Coding Exercises (30% of Unit 1 )
   1. Uploading the exercise from Week 1, this is worth **10%** of your final grade for this unit.
   2. Uploading the exercise from Week 3, this is worth **20%** of your final grade for this unit.
3. Software Design and Problem Solving Exercises (20% of Unit 1 )
   1. Uploading the UML exercise from Week 5, this is worth **10%** of your final grade for this unit.
   2. Uploading your Darts Plan from week 6, this is worth **10%** of your final grade for this unit.

Your task for Unit 2 is described over the rest of this document.

**4 Assessment Overview - Unit 2 (worth 50% of the module)**

The assessment is based on the work carried out in the tasks below. All students must complete task 1. You can choose to progress to tasks 2 and 3 if you would like to achieve a higher grade. Students who only complete Task 1 will receive a maximum grade of C+.

1. Simulating a complete game of ‘501’ on a standard darts board
2. Exploring more interesting game -play strategies taking account of the state of the match and the opponent’s score
3. Creating a graphical front end

**4.1 The problem**: A standard darts game of 501 is played on a regulation dartboard with numbers in sequence from the top:

20, 1, 18, 4, 13, 6, 10, 15, 2, 17, 3, 19, 7, 16, 8, 11, 14, 9, 12, 5, 20.

There is a “doubles” ring around the outside of the board and a “trebles” ring between the double and the bull which itself is split into an outer, worth 25, and an inner, worth 50 as shown in Figure 1 below. A dart that lands in the double or treble ring counts double or treble the segment score. For more details of dart games, please see the link https://www.darts501.com/Games.html

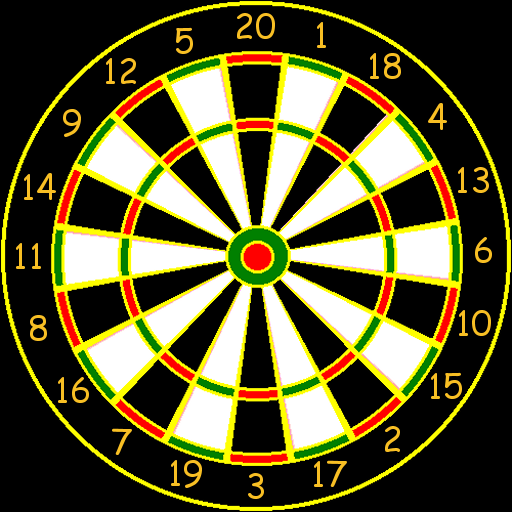


Figure 1 Dart Board

Each player has three darts each and take turns to throw at the board. Each player starts with a score of 501 and in order to win, they must have a final score of 0, achieved by landing their final dart at either a double or the bullseye. That is, their final score of 0 must have been immediately preceded by an even score of either 2, 4, 6, … , 24, 26, … , 38, 40, or, 50. For any other score, which would have led to a total of less than two, then the entire final set of darts is discounted.

To be clear, with a score of 60 remaining a player may score 20, 20, 1 leaving 19 but may not score 20, 20, 19 leaving 1. For such a throw his / her score would remain 60.

[NB in practice a player might use this to his or her advantage. For example with 32 and three darts - a very advantageous position - the player might throw 16 (missing the double) and then 11 (missing double 8). With 5 left and only one dart the options are 1 for double 2 or 3 for double 1. Neither of these is better, or faster, than deliberately busting and a return to 32 remaining and three darts for the next throw. The score is bust if a player reduces the score to 1 or goes below zero, that turn ends immediately and the score is returned to what it was at the start of that turn].

In the World Championships the final two players play a match of thirteen “sets” and each set is the best of five games (i.e. first to three). The players throw “nearest the bull” i.e. a 50:50 game, to see who throws first for game one. Thereafter, the players alternate in throwing first - as with service in tennis. This is a non-interactive game, i.e. the computer should calculate the dart score and no input is required from the user to ‘throw the dart’. Your program should generate the results as a simulation, i.e. computed over a large number of repetitions; this is known as the Monte Carlo method.

We will provide you with four functions to:

1. throw at a treble(),
2. throw for a double()
3. throw for a single(), including the outer, 25, and
4. throw for the bull().

**4.1.1 Task 1: Simple Solution**

If Joe and Sid played against each other in the final of the World Championships and if Sid threw first in game one, by what score in sets would he expect to win?

**Strategy:** For the previous 301 game that we worked on in the **practical class in Week 6** there was no strategy involved. Each player simply aimed for the optimal target with each throw. We suggest you use the same approach here. Just keep it simple and leave until task 2 the question of which route to take for a given game situation.

**What you should submit for this task:**

* One compulsory form (see end of this document).
* Pseudocode and UML diagram
* Results (see below, sample output)
* A short report (< 1 page) summarising your approach and a reflection on the benefits of object orientation in comparison with procedural programs from semester 1
* Source code (with comments)
* A working .exe file.

**Sample Output (using mock data)**

Most likely result: **Sid wins 7:4** computed over 10,000 matches with results:

# Joe : Sid Frequency

7 : 6 4%

6 : 7 8%

5 : 7 18%

**4 : 7 34%**

3 : 7 22%

2 : 7 9%

1 : 7 5%

**4.1.2 Task 2: Enhanced Strategy**

This task is a refinement of task 1 but now focusing on optimal strategy using some basic artificial intelligence. This will involve a collection of factors. Some examples below:

1. most strong players prefer to finish on either double 20 or double 16 (as a miss in the single still leaves a double) and so focus the later scoring throws accordingly.
2. there are some special cases though, for example 74. Best choice now may be treble 14 since success leaves 32 (as above) and failure is likely to score the single leaving 60.
3. not all scores can be achieved in three darts and it would certainly be important to calculate and store all scores which are one dart finishes (2, 4, … 38, 40, 50); all two dart finishes and all three dart finishes. This affects the focusing phase as with, say, 227 and just one dart remaining only a treble 19 will leave a finish at the next turn (170 = T20 + T20 + Bull).
4. Since both our players are very strong each will know that if his opponent has a finish at his next turn there is a great danger that he will succeed. Consequently every effort must be made to get out quickly – even if this involves a more risky route. So, for example, with only two darts left and a score of 107 the only chance is treble 19 and Bull but with plenty of time (darts) remaining a more normal route might be treble 19, single 18, double 16.

**Strategy:** Create a revised version of your 501 game for Task 1 as described above, including a more subtle choice of targets based on a consideration of the game situation.

If Joe adopted your enhanced strategy how much would this improve his chances against Sid playing a simple game as in Task 1?

**What you should submit for this task:**

* One compulsory form (see end of this document).
* Pseudocode and UML diagram
* Results (see above, sample output)
* A short report (< 1 page) summarising your approach, the artificial intelligence features you are using and a reflection on the benefits of object orientation in comparison with procedural programs from semester 1
* Source code (with comments)
* A working .exe file.

**4.1.3 Task 3: Interactive Game**

Rather than looking at statistical output of the machine playing on behalf of both Joe and Sid, it might be nice to play the game yourself. If it were you playing in the final with the same skill of Sid you could choose your dart selection strategy in any way you chose. You would be able to see how Sid was getting on in the current game and choose your strategy accordingly. For example, after two darts and with 50 left and only one dart remaining you almost have to try for the bull if Sid could finish on the next turn. However, if Sid’s score is greater than 170 (no opportunity to finish) you could try the more normal route of single 18 to leave double 16 for next time.

You might even be able to answer the question:

How would you have got on in the World Championship match with Sid?

Of course you could create your interactive game with a very simple text interface, for example, T20, S18, B etc. to indicate your choice of target and it would suffice to have textual output – the value of each throw and the updated score remaining but text alone would be a little dull.

**You should still complete Task 1 and be able to show the average game wins for two simulated players.**

**Strategy:** Design an interactive version of the 501 game.

**What you should submit for this task:**

* One compulsory form (see end of this document).
* Pseudocode and UML diagram
* Results - how playable is the game?
* A short report (< 1 page) summarising your approach, artificial intelligence, the interaction and a reflection on the benefits of object orientation in comparison with procedural programs from semester 1
* Source code (with comments)
* A working .exe file.

**Submission**

**Part 1:** You need to submit your work to My Learning Splace. You should submit your report in pdf form including the pseudocode and a compressed zip file of the programs, making sure that the executables run directly without the need of the compiler environment.

The final date for submitting the work is 23rd April at 23:59For the submission rules, refer to the academic regulations.

**There is no longer a demo as part of this assessment. However, student MUST carefully follow the guidance on the form at the end of this document.**

**4.1 How you will be assessed**

Your electronic submission to My Learning Space is used to demonstrate your achievements to the moderator and external examiner.

The grade will be calculated based on the complexity of your solutions, your programming approach, the report and your understanding of the programming approach that you have adopted.

**Grading Criteria**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Grade** | **C++ Knowledge** | **Programming** | | **Report writing and formatting** |
| A/A+ | An excellent grasp of knowledge and understanding of C++ syntax, program structure, and object orientation. | Excellent programming quality, following a consistent style and making use of generally accepted best practices during implementation. | | Excellent formatting and structure, with clear flow of information between sections. Excellent use of language with correct grammar and spelling. |
| B/B+ | A very good grasp of knowledge and understanding of C++ syntax, program structure, and object orientation. | Very good programming quality, following a consistent style. | | Very good formatting and structure, with good flow of information between sections. Excellent use of language with correct grammar and spelling. |
| C/C+ | A good level of knowledge and understanding of C++ syntax, program structure, and object orientation. | Good programming quality, with some minor inconsistencies. | | Good formatting and structure with reasonable flow of information between sections. |
| DD+ | Satisfactory knowledge and understanding of C++ syntax, program structure, and object orientation. | Satisfactory programming quality, with substantial scope for improvement. | | Satisfactory formatting and structure but lacking coherence of sections. |
| MF | Marginally unsatisfactory knowledge and understanding of C++ syntax, program structure, and object orientation. | Marginally unsatisfactory programming quality. | | Marginally unsatisfactory formatting and structure. |
| F | | | Performance well below the threshold level, with only limited evidence of achievement | |
| NS | | | There is no submission, or the submission contains no relevant material | |

Unit 2 – Darts Assignment

Please complete all the pages on this form and upload it along with your .exe, code and report. This form **must** **not be zipped.**

Please 🗸 which tasks you have completed:

|  |  |
| --- | --- |
| Task 1 | 🗸 |
| Task 2 |  |
| Task 3 |  |

Files for uploading – these must be checked off before uploading

|  |  |
| --- | --- |
|  | 🗸 **to indicate these have been included** |
| Code – **must be compressed as a .zip** |  |
| .exe – **this must be included in the .zip** |  |
| Report - **.pdf must be uploaded separately from the .zip** |  |

Please answer the following questions

|  |  |
| --- | --- |
| **Questions** |  |
| Does the code compile without syntax errors? If no explain what the problems are and how you’ve attempted to resolve it. |  |
| How many matches does your program simulate? |  |
| Who plays first? | Joe/Sid/user decides **\*delete as applicable** |
| What percentage accuracy have you assigned to the players? |  |

# Task 1

Explain how your code calculates the frequency across all of your simulations. If no frequency is calculated explain how you have modified your code from the 301 exercise. Please include a screen shot of the output.

|  |
| --- |
|  |

# Task 2 – if attempted/completed

Explain how you have developed your solution beyond the basic algorithm in task 1. How does a player now decide on the best target to aim for? What challenges did this present? Please include a screen shot of the frequency output.

|  |
| --- |
|  |

# Task 3 – if attempted/completed

Explain how you developed the game beyond the basic interface in task 1. How do users choose what to aim for? How did you design and develop the interface? Please include some screen shots of the user interaction.

|  |
| --- |
|  |

# Code – for each class please copy this page and paste screenshots of the code in.

|  |
| --- |
| **Class Name:** |
| **.cpp** |
| **.h** |