Department of Computer Science

Assessed Coursework

**Module Title**: Blockchain Computing

**Module Code**: CS3BC20

**Lecturer responsible**: Atta Badii

**Type of Assignment**: Coursework

**Individual/group Assignment**: Individual

**Weighting of the Assignment:** 30%

**Page limit/Word count:**  5

**Expected time spent for this assignment: ~** 25 Hrs.

**Items to be submitted**: 1 report plus link to code made accessible to assessors; submitted via BB.

**Work to be submitted on-line via Blackboard Learn by:**  12:00 noon, Friday 19th March 2021.

**Work will be marked and returned by** 15 working days after the date of submission.

## NOTES

By submitting this work, you are certifying that it is all your sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work except where explicitly the works of others have been acknowledged, quoted, and referenced. You understand that failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly. The University’s Statement of Academic Misconduct is available on the University web pages.

If your work is submitted after the deadline, *10%* of the maximum possible mark will be deducted for *each* working day (or part of) it is late. A mark of zero will be awarded if your work is submitted more than 5 working days late. You are strongly recommended to hand work in by the deadline as a late submission on one piece of work can impact on other work.

If you believe that you have a valid reason for failing to meet a deadline then you should complete an Extenuating Circumstances form and submit it to the Student Support Centre *before* the deadline, or as soon as is practicable afterwards, explaining why.

**Assessment classification**

|  |
| --- |
| **First Class (>= 70%)** |
| Outstanding quality correct and highly efficient solution with excellent analysis and rationalisation of methods used and tested with the approach to validation of correctness and efficiency thoroughly justified; excellent quality and presentation of the technical report in terms of clarity, coherence, and other aspects of the presentation such as English grammar. |
| **Upper Second (60-69%)** |
| High quality correct and efficient solution with good analysis and rationalisation of methods used and tested with the approach to validation of correctness and efficiency well-justified; high quality of presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar . |
| **Lower Second (50-59%)** |
| Good quality correct and moderately efficient solution with reasonable analysis and rationalisation of methods used and with the approach to validation of correctness and efficiency reasonably justified; good quality and presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar. |
| **Third (40-49%)** |
| Satisfactory solutions with moderate/low quality and efficiency but adequate analysis and rationalisation of methods used and tested with the approach to validation of correctness adequately justified; satisfactory presentation of the technical report in terms of clarity, coherence, and other aspects of presentation such as English grammar but with some shortcomings. |
| **Pass (30-39%)** |
| Basic (minimally acceptable quality) solution with barely adequate analysis and rationalisation of methods used and tested with the approach to validation of correctness not fully justified; adequate quality and presentation of the technical report in terms of clarity, coherence, and other aspects of presentation but with many shortcomings. |
| **Fail (<30% )** |
| Inadequate solution with inadequate analysis and rationalisation of methods used and tested; inadequate or missing validation of correctness of the approach justified; inadequate quality and presentation of the technical report lacking in clarity, coherence, and other aspects of presentation- below the minimum acceptable standards. |
|  |

**Assignment implementation requirements**

C#, Visual Studio (v# 4.7.2) <https://visualstudio.microsoft.com/vs/community/>

Assignment.Zip (downloaded from the CS3BC20 Blackboard Practicals & Coursework folder)

<https://www.bb.reading.ac.uk/webapps/blackboard/execute/content/file?cmd=view&mode=designer&content_id=_5358337_1&course_id=_158671_1&framesetWrapped=true>

**Assignment Submission requirements**

A technical report (expected length 5 pages) describing the solutions to the attempted Assignment tasks, the methods, implementation and results plus a link to the code site made accessible to the assessors. The Deadline for submission is 12;00 Hrs Friday 19th March. Please include the following information on the front sheet of your submission

**Module Code:** CS3BC20

**Assignment report Title:** Blockchain Coursework Assignment

**Student Number (e.g. 25098635):**

**Date (when the work completed):**

**Actual hrs spent for the assignment:**

**Assignment description**

**Practical Guided Exercises and Assignment**

This integrated Practicals & Coursework Assignment set for this module is intended to enable every student to implement a basic offline Blockchain. The practicals are structured to allow guided step-by-step development of the required components for the basic blockchain implementation.

It is expected that you will complete and document the first 5 parts of the Coursework Assignment as a structured sequence of 5 exercise steps as supported through step-by-step lab demonstrations as illustrated in the practicals support document which is called Parctical\_Exercises & Coursework\_Support.doc and that you can download from:

<https://www.bb.reading.ac.uk/webapps/blackboard/execute/content/file?cmd=view&mode=designer&content_id=_5358337_1&course_id=_158671_1&framesetWrapped=true>

The assignment task (part 6) is based on the application you will have built through the first 5 parts of practical lab exercises. For your submission you will submit your solution for the first 5 parts as worked out through demonstrated lab sessions plus your solution to the three tasks out of 4 of the following 4 tasks that form part 6.

To earn full marks for **implementation** in Part 6 of this coursework you only need to submit correct implementations of **three** out of **four** of these assignment tasks. As you can see in the following marking scheme table, this part 6 carries 35 marks overall, 15 for implementation and 20 for the report. You can earn the full 20 marks for the report section if you only implement three sections. However you may be able to pick up lost marks due to incomplete implementation of any three if you choose to implement all four, although in any case, for part 6, one cannot obtain more than the maximum allowed marks of 35. Research and reference to other Blockchain implementations will be valued highly for the part 6 tasks; especially if you incorporate their logic into your solution.

Marks for the assignment are allocated as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Coursework Parts** | **Marks for Evidence of implementation** | **Marks for the Report** | **Total**  **Marks** |
| Part 1 – Project Setup | 3 | 2 | **5** |
| Part 2 – Blocks and the Blockchain | 5 | 5 | **10** |
| Part 3 – Transactions and Digital Signatures | 5 | 5 | **10** |
| Part 4 – Consensus Algorithms (Proof-of-Work) | 10 | 10 | 20 |
| Part 5 - Validation | 10 | 10 | **20** |
| Part 6 – Assignment Tasks | 15 | 20 | **35** |
|  | 48 | 52 | **(100)** |
|  | | | |

For each part in this coursework assignment, you are expected to submit evidence of implementation alongside a report which details your understanding of the steps.

**The 4 tasks in Part 6 of this assignment are as follows:**

**Task 1: Extending the Proof-of-Work algorithm t**o parallelise this task such that multiple threads are in use. Currently when a node performs Proof-of-Work to find a hash to satisfy the difficulty level, only one core in the CPU is working, the rest are idle. For the report, document the implementation and prove that there is a performance efficiency gain when using multiple threads as opposed to one

**Guidance for this task:**

Take multiple samples of mining times and compare them; make use of

*(system.Diagnostics.Stopwatch*).

Do not forget to consider the fact that threads may repeat work (hashing the same data and creating hashes that have already been generated). Provide a solution that avoids such duplicated work and explain in the report how it prevents work from being repeated.

**Advice:** The following C# resources may help implementation: [Callbacks/Delegates](https://docs.microsoft.com/en-us/dotnet/api/system.func-2?view=net-5.0) and [Threading](https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread?view=net-5.0). If you have problems updating the UI using threads the following line may be useful:

textBox1.Invoke(new Action(() => textBox1.Text += message))

**Task 2: Adjusting the Difficulty Level in Proof-of-Work**

As previously mentioned in the practicals support document, the difficulty level was a static type so far and as such did not change for each new cycle of block mining. In state-of the art Blockchain (Proof-of-Work) implementations a dynamic difficulty level is used. In Proof-of-Work crypto-currencies ‘Block Time’ is considered the average amount of time required until the next Block is added to the chain. In Bitcoin this is 10 minutes and in Ethereum this is 10-20 seconds.

For this task please decide your own ‘Block time’ (with justification in the report) and implement an adaptive difficulty level algorithm. Prove that the implementation works in the report and discuss how you developed your solution for your implementation.

**Guidance for this task:**

In our current implementation increasing the difficulty by one would increase the amount of work by a factor of 16. This is not suitable for dynamic difficulty level setting. You would need to consider another approach; you may wish to review the existing approaches as adopted by others.

In your report, please detail how you developed your implemented dynamic difficulty level and why you

transactions pool. They may wish to be altruistic and pick the transactions that have been waiting the longest. They may wish to be greedy and pick the transactions with the largest fees. They may wish to be unpredictable and pick entirely randomly. They may have their own transactions pending and choose to pick up transactions involving their address first. For this task please implement a setting within the UI, as has been developed during the practical sessions, to enable the node to decide how it wishes to pick transactions from the pool. Include “Greedy” (highest fee first), “Altruistic” (longest wait first), “Random” and “Address Preference”.

**Guidance for this Task:**

In your report discuss what you believe would be the optimal setting to choose and why. Also discuss how to choose to implement the settings and provide evidence that each setting works.

**Task4:** You are welcome, within the scope of this work, to come up with your own extension/ modification of the application developed through part 1-5 of the practical sessions. Such an extension as you may design and implement for this task 4 cannot be trivial and should need an amount of effort comparable to what is needed to complete each of the first 3 tasks listed above.

**Guidance for this Task:**

Some ideas for this task 4, for example, include:

* The implementation of a different consensus algorithm (Proof-of-Stake),
* Creating multiple nodes running in a local network, automating the generation of transactions,
* Smart contracts (if you are really ambitious).

You will need to justify your choice to add any extension to the application, document your implementation and provide proof that it works as intended.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Detailed Marking Scheme & Feedback follows on the next page**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CS3BC20 Marking Scheme & Feedback Sheet** | | | | | | |
| **Aspect** | **Description** | **Potential Marks Rewarded** | | **Specific Marking Criteria, to serve also as Specific Feedback Indicator** | | |
|  |  | **Implementation** | **Documentation** | |  | |
| **Practical 1 - Project Setup** | |  |  | |  | |
| Customising the User-Interface | Addition of Buttons and Text Boxes | 1 | 1 | | Windows Forms and User Interface design | |
| Event Handlers | Implementation of Event Handler | 1 | 1 | | Handling user input | |
| Setting-up and Running the Code | Basic "Hello World" Program | 1 | 0 | |  | |
|  |  | **3** | **2** | |  | |
| **Practical 2 - Blocks and Blockchain** | |  |  | |  | |
| Block and Blockchain Class Structure | Appropriate variables and data-types including list data-structure | 1 | 1 | | Data-type justifications | |
| Instantiation of a new Blockchain | Object definition and initialisation in Blockchain app | 1 | 1 | | Class hierarchy description | |
| Genesis Block Creation | Necessary constructor modifications | 1 | 1 | | Special properties of Genesis blocks | |
| Hashing | Hashing the entire block using the SHA256 algorithm | 1 | 2 | | Description of hashing and hash properties | |
| Printing Blocks | Outputting hashes as hexadecimal strings in the UI | 1 | 0 | |  | |
|  |  | **5** | **5** | |  | |
| **Practical 3 - Transactions and Digital Signatures** | |  |  | |  | |
| Wallet Creation | Asymmetric key generation and UI adaptation | 1 | 2 | | Key usage and mathematical relationship/properties | |
| Setting up Transactions | Transaction class implementation - Variables and constructor | 1 | 0 | |  | |
| Digital Signature Creation | Signing the Hash using Senders Private Key | 1 | 1 | | Use in authentication of transactions | |
| Processing Transactions | Generate a transaction and printing the data | 1 | 0 | |  | |
| Transaction Pools | Implemented as a list of "pending" transactions | 1 | 2 | | Creating and managing Transaction Pools in Blockchains | |
|  |  | **5** | **5** | |  | |
| **Practical 4 - Consensus Algorithms (Proof-of-Work)** | |  |  | |  | |
| Generating new Blocks | Adding "Empty" Blocks to a Blockchain | 2 | 2 | | Blockchain-Block relationship | |
| Adding transactions into Blocks | Transaction Lists | 2 | 2 | | Block composition | |
| Proof-of-Work | Algorithm Implementation | 2 | 2 | | Properties, Advantages and Disadvantages etc. | |
| Nonce Generation | Random Number Generation | 1 | 1 | | Requirement for nonce in Blocks | |
| Difficulty Level | Value selection and checking | 1 | 1 | | Justification of value selected | |
| Rewards and Fees | Coinbase configuration | 2 | 2 | | Mining and Incentives: Driving transactions | |
|  |  | **10** | **10** | |  | |
| **Practical 5 - Validation** | |  |  | |  | |
| Validating the Blockchain structure | Block Coherence and contiguity checks | 2 | 2 | | | How trustability is achieved |
| Checking and Validating Balances | Ledger Tracing | 2 | 2 | | | Double spend prevention |
| Validating Blocks and Merkle Root | Implementing Merkle Root Algorithm - Combining Hashes | 2 | 2 | | | Merkle root properties and benefits |
| Validating Transactions | Checking digital signatures | 2 | 2 | | | Authenticity and Integrity achieved as a result of usage |
| Testing the Validation | Verification | 2 | 2 | | | Incorporation of "rules" |
|  |  | **10** | **10** | | |  |
|  |  | **33** | **32** | | | **65** |
| **Assignment Tasks** | |  |  | | |  |
| **Task 1 - Extending Proof-of-Work** | |  |  | | |  |
| Multi-threading | Callbacks/delegates and threading | 3 | 3 | | | Increasing the rate in which nodes mine blocks |
| e-Nonce | Additional nonce generation | 1 | 1 | | | Overcoming "Duplication of work" in parallelised systems |
| Sampling | Comparative study | 1 | 3 | | | Performance comparisons |
|  |  | **5** | **7** | | |  |
| **Task 2 - Adjusting Difficulty Level for Proof-of-Work** | |  |  | | |  |
| Block Time Measurement | Calculation of "Block Time" | 1 | 1 | | | Using "Block Time" as a metric |
| Adaptive Difficulty | Adaptive Difficulty Algorithm Implementation | 3 | 3 | | | Evidence of background reading |
| Block Time Selection | Adaptivity etc. | 1 | 3 | | | Justification of design and implementation |
|  |  | **5** | **7** | | |  |
| **Task 3 - Implementing Alternative Mining Preference Settings** | |  |  | | |  |
| Pool Adaptation |  | 1 | 3 | | | Potential use case for each preference |
| Greedy | Highest Fee | 1 | 1 | | |  |
| Altruistic | Longest Wait | 1 | 1 | | |  |
| Random | Random Selection | 1 | 1 | | |  |
| Address Preference | Owner | 1 | 1 | | |  |
|  |  | **5** | **7** | | |  |
| **Task 4 - Other Extension or Modification** | |  |  | | |  |
| Other | Alternate consensus algorithm implementation, networking, smart contracts etc. | 5 | 7 | | | Design and Implementation justification |
|  |  | **5** | **7** | | |  |
|  |  | **15** | **20** | | | **35** |
|  |  |  |  | | |  |
|  |  |  |  | | | **100** |