Student ID	29021144
Username	ih021144
Name	Anand Patel
VC	https://gitlab.act.reading.ac.uk/ih021144/29021144CS3BC20

Aspect	Description	Max. Implementation	Max. Documentation	Notes	Rewarded Implementation	Rewarded Documentation	
Practical 1 - Project Setup							
Customising the User-Interface	Addition of Buttons and Text Boxes	1	1	Windows Forms and User Interface design	1	1	
Event Handlers	Implementation of Event Handler	1	1	Handling user input	1	1	
Setting-up and Running the Code	Basic "Hello World" Program	1	0	Evidence of functionality	1	-	
		3	**2**		**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	1	1	Block fundamental unit of blockchains
Instantiation of a new Blockchain	Object definition and initialisation in Blockchain app	1	1	Class hierarchy description	1	1	Blockchain -> List<Block>
Genesis Block Creation	Necessary constructor modifications	1	1	Special properties of Genesis blocks	1	0	Not specified explicitly
Hashing	Hashing the entire block using the SHA256 algorithm	1	2	Description of hashing and hash properties	1	1	SHA-256 identified and utilized effectively; Hash and other cryptographic functions confused slightly (e.g. hashing is uni-directional not asymmetric); Usage for immutability stated
Printing Blocks	Outputting hashes as hexadecimal strings in the UI	1	0	Evidence of functionality	1	-	
		5	**5**		**5**	**3**	
Practical 3 - Transactions and Digital Signatures							
Wallet Creation	Asymmetric key generation and UI adaptation	1	2	Key usage and mathematical relationship/properties	1	2	ECDSA utilized effectively; Key scope and usage clearly differentiated
Setting up Transactions	Transaction class implementation - Variables and constructor	1	0	Evidence of functionality	1	-	
Digital Signature Creation	Signing the Hash using Senders Private Key	1	1	Use in authentication of transactions	1	1	Authenticates the transaction
Processing Transactions	Generate a transaction and printing the data	1	0	Evidence of functionality	1	-	
Transaction Pools	Implemented as a list of "pending" transactions	1	2	Creating and managing Transaction Pools in Blockchains	1	2	Explored in relation to bitcoin implementation
		5	**5**		**5**	**5**	
Practical 4 - Consensus Algorithms (Proof-of-Work)							
Generating new Blocks	Adding "Empty" Blocks to a Blockchain	2	2	Blockchain-Block relationship	2	2	
Adding transactions into Blocks	Transaction Lists	2	2	Block composition	2	2	Block -> List<Transaction>
Proof-of-Work	Algorithm Implementation	2	2	Properties, Advantages and Disadvantages etc.	2	2	+ Facilitates decentralization - Delays to transactions and resource requirements
Nonce Generation	Random Number Generation	1	1	Requirement for nonce in Blocks	1	1	Ensures fair and competitive mining of blocks
Difficulty Level	Value selection and checking	1	1	Justification of value selected	1	1	Default difficulty of 4 set resulting in block times less than 10s for testing
Rewards and Fees	Coinbase configuration	2	2	Mining and Incentives: Driving transactions	2	1	Briefly explored and justified
		10	**10**		**10**	**9**	
Practical 5 - Validation							
Validating the Blockchain structure	Block Coherence and contiguity checks	2	2	How trustability is achieved	2	2	Structure prevents tampering
Checking and Validating Balances	Ledger Tracing	2	2	Double spend prevention	2	1	
Validating Blocks and Merkle Root	Implementing Merkle Root Algorithm - Combining Hashes	2	2	Merkle root properties and benefits	2	2	Efficient storage and validation of a blocks transactions
Validating Transactions	Checking digital signatures	2	2	Authenticity and Integrity achieved as a result of usage	2	1	
Testing the Validation	Verification	2	2	Incorporation of "rules"	2	2	
		10	**10**		**10**	**8**	
		33	**32**		**65**	**33**	**27**	**60**
Assignment Tasks							
Task 1 - Extending Proof-of-Work							
Multi-threading	Callbacks/delegates and threading	3	3	Increasing the rate in which nodes mine blocks	2	2	Multi-threading successfully incorporated to improve efficiency of mining; Cancellation token used to terminate threads when successful hash is found
e-Nonce	Additional nonce generation	1	1	Overcoming "Duplication of work" in parallelised systems	1	1	Uses thread ID to initialize nonce (no guarantee two thread ids will not be similar resulting in overlap - best to include as separate variable in addition to traditional nonce in hash input)
Sampling	Comparative study	1	3	Performance comparisons	1	3	Comprehensive study conducted, presented and analysed
		5	**7**		**4**	**6**	
Task 2 - Adjusting Difficulty Level for Proof-of-Work							
Block Time Measurement	Calculation of "Block Time"	1	1	Using "Block Time" as a metric	1	1	10s target block time
Adaptive Difficulty	Adaptive Difficulty Algorithm Implementation	3	3	Evidence of background reading	1	1	Moving average solution calculated on block-by-block basis (consider sliding-window approach and thresholding to prevent continuous adjustment/fluctuation less sensitive to anomolies); Utilizes standard puzzle with large step in difficult - potential to explore alternate puzzles
Block Time Selection	Adaptivity etc.	1	3	Justification of design and implementation	1	2	Requirement well justified
		5	**7**		**3**	**4**	
Task 3 - Implementing Alternative Mining Preference Settings							
Pool Adaptation		1	3		1	2	
Greedy	Highest Fee	1	1		1	1	
Altruistic	Longest Wait	1	1	Potential use case for each preference	1	1	
Random	Random Selection	1	1		1	1	
Address Preference	Owner	1	1		1	1	
		5	**7**		**5**	**6**	
Task 4 - Other Extension or Modification							
Other	Alternate consensus algorithm implementation, networking, smart contracts	5	7	Design and Implementation justification	3	3	Proof-of-stake explore and implemented
		5	**7**		**3**	**3**	
		15	**20**		**35**	**15**	**19**	**34**
					100			**94**