Process Management & Distributed Computing Criteria Assessment Sheet (*Weighting 30%)*

Student Name(s): **Student Number(s):**

Criteria / Objective	Performance Level					Mark
	7	**6**	**5**	**4-3**	**2-1**	
Implementation						
Functionality Implements the required functionality as stated in the Assignment Specification Document. NOTE: If the program does not compile on the command line in the Linux environment used during the weekly practicals your submission will be heavily penalised. Implementations written in Visual Studio or other IDE's that do not compile on the Linux command line will receive a mark of zero (0).	• Server implements a thread pool - max of 10 concurrent client connections. • Synchronisation primitives for shared data structures • Authentication implemented • Text file(s) tokenised correctly • Client displays account information as per specifications • Signal handling • ATM menus as per specifications • Command line & default parameters	• Server creates new thread for each client – max 10 clients • Synchronisation primitives for shared data structures • Authentication implemented • Text files tokenised correctly • Client displays account information as per specifications • Signal handling • ATM menus as per specifications • Command line & default parameters	• Single-thread client implementation only • Authentication implemented • Text files generally tokenised correctly • Client displays account information mostly as per specifications • Signal handling mostly correct • ATM menus generally correct however lacks some functionality • Command line & default parameters • Implements most of the required functionality	• Correctly implements some of the functionalities as stated in assignment specifications • The program runs but contains some run time errors during execution • Signal handling not implemented • No command line arguments and/or default values • Not all of menu options implemented • Client displays incorrect account information &/or not as per specifications • Unnecessary file I/O overhead	• Implements only small part of functionality as stated in assignment specifications • Contains serious run time errors • Application not menu driven • Application is not implemented using BSD sockets • Limited, corrupted or no communication between server and client • Is not a valid C program &/or does not compile on Linux command line	
	60-75 marks	45-59 marks	35-44 marks	25-34 Marks	0-24 marks	**/75**
	Comments:					

Criteria / Objective	Performance Level					Mark
	7	6	5	4-3	2-1	
Code Quality						
Implementation uses efficient data structures with minimal code duplication. Code is structured logically and efficiently, sensible variable and function names. All functions are commented (including pre and post conditions). Sensible use of header files. Use of thread safe functions. Dynamic memory, threads, sockets and files are handled efficiently during program execution and program termination. Efficient parameter passing between functions. No violation of constraints with respect to word pairs being visible to client.	• Efficient use of ADTs to represent required data structures • Code logically structured with minimal code duplication • Endianness addressed • Professionally commented • Dynamic memory allocated is freed. • Threads, sockets and files managed efficiently • No run time errors/compilation warnings • Data synchronisation • No magic numbers	• Appropriate used of ADTs for implementing functionality has been used in majority of code. • Sensible Synchronisation primitives mostly implemented • Source code is mostly efficient with minimal code duplication • Majority of code is commented professionally • Dynamic memory and file I/O managed reasonably well • Sensible use of global variables only where appropriate	• ADTs implemented appropriately in most cases • Some data synchronisation implemented • Synchronisation primitives appropriate in most cases • Code commenting is only adequate • Use of magic numbers rather than using defines &/or constants • Some inappropriate use of global variables • Not all dynamic memory is released • File I/O and thread management is not overly efficient • Code structure reasonable however some code duplication	• ADTs implemented appropriately in some cases • Inefficient use of synchronisation primitives • Race conditions, busy waits and deadlocks may occur • Functions are overly long and/or there is significant code duplication • Violation of constraints of handling of text files as per assignment specifications • Poor variable and/or function names • Source code is inefficient in 50% or more of implementation • Minimal to no commenting	• Poorly implemented ADTs • Large amount of straight line code rather than implementing sensible functions • Large amount of code duplication • Minimal to no synchronisation primitives • Deadlocks, starvation, race conditions may or do occur • Code does not compile at Linux command line • Frequent run time errors • Code is generally very poorly structured • No commenting	
	18-20 marks	15-17 marks	10-14 marks	5-9 Marks	0-4 marks	**/20**
	Comments:					

Criteria / Objective	Performance Level					Mark
	7	6	5	4-3	2-1	
Assignment Report						
Report consists of: 1. Statement of Completeness 2. Description of Data Structures 3. Description of synchronisation primitives 4. Each student's contribution to submission NOTE: Please include instructions for compiling both the Server and the Client at the command line in a separate text file If both students' contribution is ***not*** roughly equal each student must sign document to signify agreement in disclosure.	• Statement of Completeness indicates each task attempted including any deviations from the assignment specification, and/or problems and deficiencies in the solution. • Short concise description of ADTs implemented with valid justification of choices made • Short concise description of the synchronisation primitives with valid justification of choices made • Short statement of each student's contribution (if each student contributes equally simply state this explicitly)	• Statement of Completeness is mostly relevant and truthful • Discussion of the ADTs implemented is mostly complete and justification is generally reasonable • Description of the synchronisation primitives implemented is mostly complete with reasonable justification • Short statement of each student's contribution (if each student contributes equally simply state this explicitly)	• Statement of Completeness is fairly relevant and truthful • Discussion of the ADTs implemented is acceptable but lacks details and/or justification for choices • Description of the synchronisation primitives implemented is acceptable but lacks details and/or justification for choices • Statement of each student's contribution is included but not clear	• Statement of Completeness includes relevant information but is incomplete with significant omission of deficiencies in solution • Discussion of the ADTs implemented is attempted but incomplete, not justified or poorly justified • Description of the synchronisation primitives implemented is attempted but incomplete, not justified or poorly justified • No disclosure of each student's contribution	• No Statement of Completeness • No discussion of ADTs implemented • No discussion of synchronisation primitives • No report included in submission • No disclosure of each student's contribution	
	5 marks	4 marks	3 marks	1-2 Marks	0 marks	**/5**
	Comments:					