Core Undergraduate Computer Science Courses

Please, indicate below the grades that you obtained in the following courses (N/A if you did not take that course). If your university differentiates between "theory course" and "laboratory", please indicate only the "theory" part. See the next page for the description of these courses at the University of Ottawa.

Student Name: Prajit Bhumitsinh Zala

COURSE	Your Grade	Max Possible Grade
"Discrete Structures" (or "Discrete Math"):	B+	A+
"Data Structures" (or "Data Structures and Algorithms 1"):	B	A+
"Algorithms" (or "Data Structures and Algorithms 2"):	A+	A+
"Formal Languages " (or " Theory of Computing" or "Automata Theory"):	A	A+
"Databases":	B+	A+
"Computer Architecture":	B	A+
"Operating Systems":	C	A+
"Software Engineering" (or "Software Development"):	A+	A+

Indicative description of core CS courses (note that the description from your University might differ slightly).

Discrete Structures: Discrete structures as they apply to computer science, algorithm analysis and design. Predicate logic. Review of proof techniques; application of induction to computing problems. Graph theory applications in information technology. Program correctness, preconditions, postconditions and invariants. Analysis of recursive programs using recurrence relations. Properties of integers and basic cryptographical applications.

Data Structures: The concept of abstract data types. Simple methods of complexity analysis. Trees. The search problem: balanced trees, binary-trees, hashing. Sorting. Graphs and simple graph algorithms: traversal, minimum spanning tree. Strings and pattern matching.

Algorithms: Analysis of algorithms: worst-case analysis, complexity analysis, asymptotic notations and basic complexity classes. Algorithm design techniques: brute force, divide and conquer, dynamic programming, greedy, backtracking. Computational complexity of problems: lower bound arguments, the classes P, NP, NP-complete, dealing with NP-complete problems.

Formal Languages: Regular languages, finite automata, transition graphs Kleene's theorem. Finite automata with output. Context-free languages, derivation trees, normal form grammars, pumping lemma, pushdown automata, determinism. Decidability. Recursively enumerable languages, Turing machines, the halting problem.

Databases: Fundamental database concepts. Entity-Relationship modeling. Relational algebra and relational calculus. Relational databases. Database definition and manipulation using SQL. Embedded SQL. Functional dependencies and normalization. Introduction to physical database design. Design and implementation of a database application in a team project.

Computer Architectures: Design a digital computer to execute a given instruction set. Design of digital computers. Register transfer and microoperations. Designing the instruction set, CPU and CPU control. Basic machine language programming. Using pipelines for CPU design. Designing the memory unit. Designing Imput-Output subsystem.

Operating Systems: Principles of operating systems. Operating systems design issues. Process management, process scheduling, concurrency issues. CPU scheduling. Memory management. Virtual memory. Mass storage systems. Input/Output system. File system. Security and protection. Examples of operating systems.

Software Engineering: Principles of software engineering: Requirements, design and testing. Review of principles of object orientation. Object oriented analysis using UML. Frameworks and APIs. Introduction to the client-server architecture. Analysis, design and programming of simple servers and clients. Introduction to user interface technology.