Data Structures and Algorithm 2, Course Project 2020

Important – Read before starting

- The deadline for completing and submitting your assignment is strictly Friday 29th May 2020 at 18:00.
- VLE may be set up to not accept late submissions meaning that you will get zero marks if your submission if late. Please plan ahead (it is recommended that you upload and verify your work a day before).
- You must complete the project completion form (shown later) and include it in your report. Submissions without the statement of completion will not be considered.
- You must complete a plagiarism declaration form and include with your submission. Submissions without the form will not be considered.
- Projects must be submitted using VLE only. Physical copies or projects (including parts of) sent by email will not be considered.
- For your convenience, a draft and final submission area will be set up in VLE. Only projects submitted in the *final* submission area will be graded. Projects submitted to the draft area will not be considered.
- It is suggested that after submitting your project, you redownload it and check it again. It is your responsibility to ensure that your upload is complete, valid, and not corrupted. You can reupload the assignment as many times as you wish within the deadline.
- Your project must be submitted in ZIP format without passwords or encryption. Project submitted in any other archiving format will not be considered.
- The total size of your ZIP file should not exceed 38 megabytes.
- Your submission should include your report in PDF format, your source code, executable file(s), and other relevant attachments.
- Do not include all your source code in the report (small snippets for discussion are allowed).
- It is expected that you submit a quality report with a proper introduction, discussion, evaluation of your work, and conclusions. Also, make sure you properly cite other people's work that you include in yours (e.g. diagrams, algorithms, etc…).
- In general, I am not concerned with which programming language you use to implement this project. However, unless you develop your artifact in BASIC, C, C++, Objective C, Swift, Go, Pascal, Java, C#, Matlab, or Python, please consult with me to make sure that I can correct it properly.
- Please provide clear instructions about how to run your program.
- This is not a group project.
- Plagiarism will not be tolerated.

Part 1 – Boolean Satisfiability

- Write a command line program that accepts Boolean expressions in Conjunctive Normal Form (CNF).
    - Use the symbols "w", "x", "y", and "z" to represent literals/variables.
    - Use the symbol "!" to represent negation.
    - Use parentheses to group clauses.
    - Use commas to separate literals in clauses.
    - Whitespace is irrelevant and should be ignored while parsing.
    - Example expressions include:

$$(w)(x,!\,y)$$
$$(x,y)(w)(!\,x,z)$$

- Implement the Davis-Putnam-Logemann-Loveland (DPLL) algorithm to determine whether the expression is satisfiable.
    - If the expression is satisfiable, your program should display a truth assignment as a proof.
    - If the expression is not satisfiable, your program should display "UNSAT".
- In your report, dedicate a section to describe the practical applications of SAT.

Part 2 – Data Compression

- Write a command line program, which:
- Accepts an ASCII text file as an input.
- The only characters allowed in the input file are **`A-Z + a-z + 0-9`**. If the file contains any other characters, an error should be shown and the program aborted.
- Implement Huffman coding to build the optimal coding tree for the contents of the file.
- Display the coding table (sorted by ASCII value) of each character in the character set. E.g.:

A: 000
B: 0011
C: 1111
… and so on

Statement of completion – MUST be included in your report

Item	Completed (Yes/No/Partial)
Part 1 – Accepted and parsed input.	
Part 1 – Implemented DPLL	
Part 2 – Implemented Huffman coding	
If partial, explain what has been done	

Marking Breakdown

Description	Marks allocated
Parsing CNF expressions	10%
Implemented DPLL	30%
Implemented Huffman coding	15%
Section about practical applications of SAT	10%
Evaluation and testing of artifacts	15%
Overall report quality	20%