UTS
UNIVERSITY OF TECHNOLOGY SYDNEY

ASSIGNMENT COVERSHEET

UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	NAME OF STUDENT(s) (PRINT CLEARLY)	STUDENT ID(s)
48430 Fundamentals of C Programming	*Wangbo Xia* *Siyu Fang* *Bo Peng* *Zhengxin Ma* *Nuojin Zhang*	13100072 13078985 13317540 13485952 13482397

STUDENT EMAIL	STUDENT CONTACT NUMBER
bo.peng-6@student.uts.edu.au wangbo.xia@student.uts.edu.au siyu.fang-1@student.uts.edu.au nuojin.zhang-1@student.uts.edu.au zhengxin.ma-1@student.uts.edu.au	

NAME OF TUTOR	TUTORIAL GROUP	DUE DATE
Guoqiang Zhang	Lab 07 Group 06	28/05/2021

ASSESSMENT ITEM NUMBER & TITLE

☑ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.
☑ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.
☑ I understand that if this assignment is submitted after the due date it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of collaboration:

Signature of student(s) Zhengxin Ma Nuojin Zhang Wangbo Xia Siyu Fang Bo Peng **Date** 28/05/2021

✂- -

ASSIGNMENT RECEIPT

To be completed by the student if a receipt is required

SUBJECT NUMBER & NAME	NAME OF TUTOR

SIGNATURE OF TUTOR	RECEIVED DATE

STYLE GUIDE for ASSIGNMENT SUBMISSION

Before submitting an assignment, you should refer to the policies and guidelines set out in the following:

- FEIT Student Guide
- UTS Library - referencing
- HELPS - English and academic literacy support
- UTS GSU - coursework assessment policy and procedures

Unless your Subject Coordinator has indicated otherwise in the Subject Outline, you must follow the instructions below for submission of assignments in the Faculty of Engineering and Information Technology.

Writing style

It is usually best to write your initial draft in the default settings of your software without formatting. Use the following guides in your writing.

Purpose and audience: use the correct genre and language style expected for the particular task.

Language: use 'plain English' for all technical writing. More information about this language style can be found at www.plainenglish.co.uk/free-guides.html.

Use spelling and grammar software tools to check your writing. Edit your document.

Standards: always use:

- Australian spelling standards (Macquarie Dictionary)
- SI (International System of Units) units of measurement
- ISO (International Organisation for Standardisation) for writing dates and times for international documents. For example **yyyy-mm-dd** or **hh-mm-ss**. However, for most applications it is more helpful to present the date in full as **26 August 2016**.

Graphics and tables should:

- be numbered
- have an appropriate heading and/or caption
- be fully labelled
- be correctly referenced.

Presentation

Unless otherwise instructed, all assignment submissions should be **word processed** using spell-check and grammar-check software. Work should be well **edited** before submission. Use the following default settings:

Page setup: set margins at no less than 20mm all around.

Paper: print on A4 bond, double-spaced and preferably double-sided, left justified.

Font: use the software default style to provide consistency. The recommended style includes:

- 10-12 pt font
- consistent formatting with a limited number of fonts
- lines no more than 60 characters (use wider margins or columns if you need to make lines shorter)

Header should include:

- your name and student number
- the title of the paper or task.

Footer should include the page number and current date.

Cover sheet and statement of originality: all work submitted for assessment must be the original work of the student(s) submitting the work. A standard faculty cover sheet (see over) must be attached to the front of the submission. Any collaboration between the submitting student and others must be declared on the cover sheet.

Referencing

All sources of information used in the preparation of your submission must be acknowledged using the Harvard system of referencing. This includes all print, video, electronic sources.

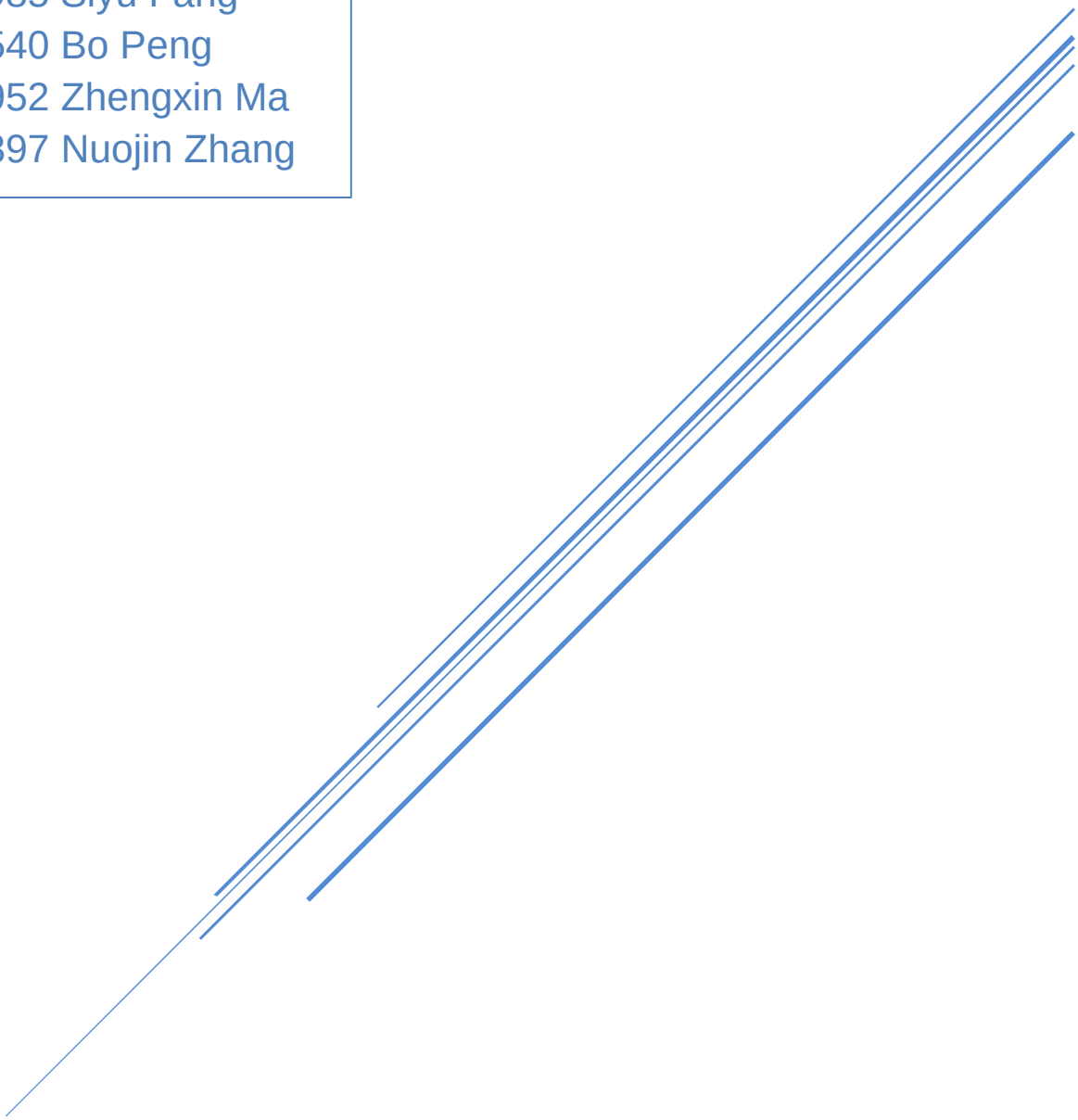
Phrases, sentences or paragraphs taken verbatim from a source must be in quotation marks and the source(s) cited using both **in-text** referencing and a **reference list**.

Plagiarism is the failure to acknowledge sources of information. You should be fully aware of the meaning of plagiarism and its consequences both to your marks, position at the university and criminal liability. The plagiarism in your assignment submissions can be assessed both in hard copy and in soft copy through software such as Turnitin.

The UTS Library and UTS HELPS (web links above) provide extensive information for students on referencing correctly to support you in avoiding plagiarism.

48430 FUNDAMENTALS OF C PROGRAMMING ASSESSMENT - GROUP 06

13100072 Wangbo Xia
13078985 Siyu Fang
13317540 Bo Peng
13485952 Zhengxin Ma
13482397 Nuojin Zhang



UTS
48430

Table of Contents

Objective

Our project aims to help train station employees save the information of a specific train such as Train Number or Departure Station. Therefore, the project can also be named the Train Information Collection System. Through the TIC system, an employee can add one or more trains into the TIC system. Also, the system can display the information of all the trains added to the system.

Project Scope

The core function of the Train Information Collection system is to allow an employee to store the information of all the trains entered into a database.txt file, and the employee can choose to download it from the system. Moreover, the TIC system can compress the database.txt file to make its size smaller, but after compression, the original database.txt file will be removed accordingly. The newly created database. comp file will be unreadable, thus, if a user wants to read the contents of the database. comp file, he/she has to decompress the database.comp file. Similarly, the database. comp will also be removed by the project automatically.

To make the database file more secure, the Train Information Collection system can let employees encrypt the database.txt. After the encryption, the original database.txt will be encrypted into database.txt. Xor if an employee chooses to use XOR to encrypt a file. Additionally, there are two methods for encrypting a file, and one is called Subtitiation, another one is called XOR. Users are allowed to choose any of these two methods for encryption.

On the other hand, since the Train Information Collection system has the encryption function the Train Information Collection system also has the decryption function, which can help users of this system decrypt and read the contents of the target file.

Program Features

1. User Interface

The project have been added a command-line interface for users to input their requests. By inputting the number required it will match the function numbers and call the corresponding function. The numbers greater than 10 or the other inputs will print ‘Invalid input’.

```
1. Add a train infomation
2. Display all trains to a destination
3. Store the train information into the database file
4. Sort train number in ascending order
5. Load the train information from the database
6. Compress the database file
7. Decompress the database file
8. Encrypt the target file
9. Decrypt the target file
10. Exit the program
Enter choice (number between 1-10)>
```

2. Database Functions

addTrain()

When this function is called, it checks if the maximum number of trains is reached. If no more trains could be added, it will inform the user and stop. Otherwise, it asks the user to input the train code, departure station, arrival station, and time. If all the information is valid, the train information will be stored in an array. Otherwise, the function will require the user to enter it again and print out ‘Invalid input’.

The function uses several different ways to prevent the user from making input errors, such as comparing the size of strings with each other, checking the length of the user's input, using a fixed-length array, counting the number of correct outputs, etc.

displayAllTrains()

The function includes two parts of the required functions, displaying and searching. It will ask the user to enter the arrival station’s name after being called, and it will display all the trains stored in the train’s array in memory with the corresponding information with the same arrival. The function uses a for loop to compare the values of the arrival station one by one. if there is no matching information in the trains array, the function will tell the user that there is no matching information. If the user enters *, then the function will display all the saved information.

saveAllTrains()

This function saves the train information in memory to a local file on the hard disk. The function first checks if the train’s array contains train elements. If no data is saved in the memory it will print ‘No Trains’. The data will be saved if the array is not empty and generate a text file to show all the information. Otherwise, the function will inform the user and stop.

sorting()

This function will sort the train number in ascending order(Skipped the letter in the front) and display the sorted number on the terminal. Also, it will relate train numbers and corresponding train information together and display them all on the terminal. Then it can use the SaveAllTrains() function to save the sorted information into the txt file formed before.

loadDatabase()

This function will load the saved information from the local file. It reads the train information from the txt file by specific input format. When the program starts, it will call this function to read from the previously saved database file if it exists.

3. Compression

The program provides a feature for users to compress and decompress a file. This feature requires a user to enter a particular text filename like database.txt. And then, the program passes the filename to

the Compress() function. Then, the database.txt starts to be compressed by the program. If the size of the database.txt is too small, the program will return an error message telling the user that the file is too small, It is not compressed. If the name for a specific text file does not exist, the program will also return an error message telling the user that the File does not exist.

Besides, the program will show the compression ratio if the program compresses a text file successfully. Meanwhile, the compression function will also return a message telling the user that the compression was successful.

In theory, the compression function can compress text files only. If the target file format does not match, the compression function will also return an error message telling the user that the format is not correct.

The original file after compression will be turned into a binary file, which is unreadable for people. In other words, the file contains binary data and cannot be displayed. To make the contents of the file after compression readable, I also introduced the decompression function.

As for the decompression, the function is responsible for decompressing the compressed file back to a text file, which is readable. Thus, the original compressed file will be removed, and there will be a text file after compression.

Both of the compression and decompression functions are based on Huffman Coding provided on Canvas. There is a folder called Compression; the Compression folder contains two fields, which are compression.h and compression.c, respectively. The compression.h is used to define relevant function prototypes, and the compression.c is used to implement those function prototypes. Also, I implemented the node of the Huffman tree in the compression.h file, which would be used for Huffman Compression Algorithm. The Compression folder could be used as an interface for compressing and decompressing a text file.

```
379 /*****************************************************************************
380  * This function used for compression
381 *****************************************************************************/
382 void compressOnly() {
383     char inFileName[1024];
384     printf("Enter the filename you wish to compress?\n");
385     scanf("%s", inFileName);
386     Compress(inFileName);
387 }
388 /*****************************************************************************
389  * This function used for decompression
390 *****************************************************************************/
391 void decompressOnly() {
392     char inFileName[1024];
393     printf("Enter the filename you wish to decompress?\n");
394     scanf("%s", inFileName);
395     Decompress(inFileName);
396 }
```

- CompressOnly() function compresses a target text file with the Huffman algorithm.
- DecompressOnly() function decompresses a text file compressed

4. Encryption

The program provides two file encryption methods: Substitution encryption and the other is XOR encryption. The encryption function is intended to improve the security and confidentiality of the file. When the user does not want the file to be made public, the encryption function can be used to make the file's content unreadable, and the decryption function can be used to decrypt the file when necessary.

The substitution cipher is a straightforward encryption algorithm. It replaces the letters according to the rules prescribed in advance. In this program, the characters in the file will be shifted backward within the ASCII printable characters.

Compared to the substitution cipher, the XOR cipher requests the user to set a password before encryption. Then, the program will do xor operation of each character in the file and the character in the key, which will produce an unreadable binary file.

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
------------------------------------
= 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
------------------------------------
= 01010111 01101001 01101011 01101001
```

XOR operation (Wikipedia 2021)

Problems Encountered

1. Compression Challenges

One of the hardest parts was understanding the Huffman tree concepts and how to use them for file compression. Fortunately, there were many example websites for Huffman Tree Algorithms on the Internet.

Another one, there was a bug while compressing a text file using Huffman Compression Algorithm. The compression algorithm would still return the error message telling a user that the compression was successful, even if the text file size was too small. The correct message produced by the program should be that the file cannot be compressed when the size of a text file is too small.

The program is supposed to delete the original database.txt file if the compression function works as expected. However, there was also a bug inside the compression function at that time, which was to cause that the function could not delete the original database.txt after successful compression.

2. Encryption Challenges

The substitution algorithm needs to specify the mapping relationship of each character and must be reversible, so the range of characters needs to be limited. Therefore, the substitution in this program can only be used to encrypt ASCII-encoded text files and cannot encrypt other types of files. On the other hand, the XOR algorithm can encrypt any file, and its security is higher than the replacement algorithm.

3. Input errors

In program design, the key issue is to consider the impact of invalid or illegal user input on program operation. In the testing phase, furthermore, the type and length of the data entered by the user may affect the regular operation of the program. Therefore, in the program, it automatically checks the data type entered by the user and discard the out-of-range input to ensure the validity of the information.

Conclusion

The project's objective is to help train station employees when they attempt to upload and save relevant train information, so the project would be named the Train Information Collection system.

There are four core functions: train information adding function, database file creating function, compression and decompression function, encryption, and decryption process, respectively.

An employee can add the information of a train via the Train Information Collection system, and the system can list out the information of all the trains on the terminal. Besides, the data can also be stored in an external text file called database.txt so that anyone with access to the Train Information Collection system can download this file.

Meanwhile, group 06 also introduced file compression and decompression functions to reduce the database file size. Furthermore, the system has added the encryption and decryption functions accordingly. Thus, the compressed file will be more secure.

Reference

En.wikipedia.org. 2021. *Huffman coding - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Huffman_coding> [Accessed 28 May 2021].

Wikipedia 2021, *XOR cipher*, viewed 27 May 2021, <https://en.wikipedia.org/wiki/XOR_cipher>

Appendix

Individual Contribution Logbook

Wangbo Xia (13100072)

- Implemented Compression and Decompression functions successfully

- Adjusted the structure of the Project
- Made a Makefile
- Wrote a draft for the group report
- Solved bugs for the group C program project

Bo Peng (13317540)

- Implemented Encryption and Decryption methods
- Built the program architecture
- Formatted the database file
- Completed error handling methods

Siyu Fang (13078985)

- Implement a function to print out the menu.
- Implement a function to input all the train's information.

Zhengxin Ma (13485952)

- Implement a function to save all trains information into a .txt file
- Implement a function to read information in the text file and load it to terminal
- Implement a function to display all information on terminal
- Adjusted the basic functions and fixed the glitches.

Nuojin Zhang (13482397)

- Implement a function to sort the train numbers in ascending order
- Implement a function to match the number inputted and call the corresponding function.
- Implement a function to search trains by entering the arrival station (In 'DisplayAllTrains()')
- Adjust some simple functions

User Journey Map

Process Stage User (Employee) Perspective

Stage	Add Information of A Train	Display the information of the train added	Store the information into an external database text file	Compress the database.txt file	Decompress the database.txt compressed	Encrypt the file compressed before	Decrypt the file	Quit from the Train Information Collection system
Goals	An employee is planning to add relevant information of a train via the Train Information Collection system	An employee can enable the terminal to display a specific train's information or the information of all the trains on the terminal via the TIC system	An employee is able to store the information of all the trains added into an external text file via the TIC system	An employee can choose to make the program compress the database.txt via the TIC system	An employee can let the program decompress the text file compressed before	An employee is also able to encrypt the database.txt file via the TIC system	An employee is able to decrypt the file via the TIC system	An employee can quit from the TIC system
Touchpoints	-Train Adding Function	-Train's Info Display Function	-Database Creating Function	-File Compression Function	-File Decompression Function	-File Encryption Function	-File Decryption Function	-Exit Function
Activities	-Adding a train's information to the TIC system	-Displaying one or more train's info on the terminal	-Save the information of all the trains into an external text file called database.txt	-Compress the database.txt file to reduce its size	-Decompress the database comp file to make it readable	-Encrypt the database.txt file to make it more secure	-Decrypt the file encrypted before	-Terminal the program on the terminal
Target User Satisfaction level	5	5	5	5	5	5	5	5