CENTENNIAL COLLEGE
School of Engineering Technology
and Applied Science

Department of Information and Communication Engineering Technology

Professor: Aime M. MBOBI, Ph.D.
Email: mmbobi@my.centennialcollege.ca

Programming 3 - Project

Group Name:

Questions		**Comments**	**Marks**
Question I			**/10**
Question II	II.1		**/5**
	II.2		**/5**
	II.3		**/5**
	II.4		**/5**
	II.5		**/5**
	II.6		**/5**
	II.7		**/5**
	II.8		**/5**
Question III	III.1		**/15**
	III.2		**/5**
	III.3		**/5**
	III.3		**/20**
	III.3		**/5**
Total			**100**

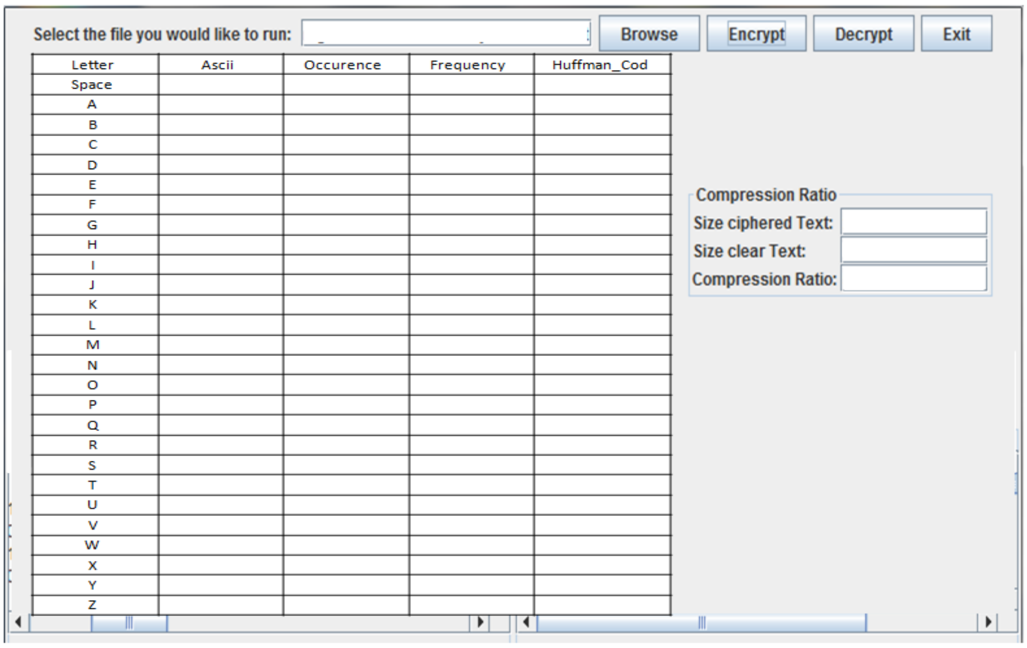
Implementation of “Substitution Compression” Algorithm

Instructions: This is a Project, therefore should be intensively commented.

Write a C# program that performs the three steps following steps Design a GUI, Builds the dictionaries, and Code and decode the Huffman Code:

I. GUI Design (10 marks)

1. Creates a graphical user interface as shown below and displays it.



Letter	Ascii	Occurence	Frequency	Huffman_Cod
Space				
A				
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				
L				
M				
N				
O				
P				
Q				
R				
S				
T				
U				
V				
W				
X				
Y				
Z				

II. Building Dictionaries (40 marks)

When the user clicks on the button `Browse`, the program:

1. Lists the directory where the file is located so that the user selects the file (we will use `IronHill.txt`). **(5 marks)**
2. Creates a first Dictionary “`Ascii`” that records the 26 alphabetical letters (Key) including the space (from A to Z plus space) along with their corresponding ASCII values (Value) **(5 marks)**
3. Creates a second dictionary named “`Occurrence`” and stores the same letters (Key) and their occurrences (Value), all of them initialized to zero **(5 marks)**
4. Reads `IronHeel.txt` file (provided) character by character and incrementally updates the occurrence of each character in “`Occurrence`” dictionary **(5 marks)**

Note: Any character different from the 26 alphabetical letters is ignored, except space, comma, colon, semi-colon that are counted and coded as "space")

5. After reading the full text, and updating "Occurrence", the program
   a. Creates a third dictionary named "`Frequency`".
   b. Calculates the frequency of each character from "`Occurrence`", and inputs each letter (Key) with its related frequency (Value) into "`Frequency`" dictionary. **(5 marks)**

   Note: The letter's frequency is the total of its occurrences divided by the sum of the letters in the text

6. Creates a fourth dictionary named "`Ordered_Frequency`" that stores each letter (Key) and its related frequencies (Value) from the highest frequency to the lowest frequency **(5 marks).**
7. Creates a fifth dictionary named "`Huffman_Code`" that stores each letter (Key) and its related Huffman code (Value) respecting the following rule **(5 marks)**

   Rule:

   a. The most appeared letter will have short coding format (less digits)
   b. The less appeared letter will have long coding format (more digits)
   c. Since the coding chart starts from the short to the long format, you will start coding from the most to the less appeared letter
8. Displays all the dictionaries in the GUI **(5 marks)**

Letter	Frequency	Coding
Most used Letter		100
2nd Most used Letter		0010
3rd Most used Letter		0011
....		1111
		1110
		1100
		1011
		1010
		0110
		0101
		11011
		01111
		01001
		01000

Letter	Frequency	Coding
		00011
		00010
		00001
		00000
		110101
		011101
		011100
		1101001
		110100011
		110100001
3rd Less used Letter		110100000
2nd Less used Letter		1101000101
Less Used Letter		11010001000

III. **Huffman Coding – Decoding (50 marks)**

When a user clicks on the button `Encrypt` the program (20 marks):

1. Builds the ciphered text by converting each characters into its `Huffman` representation (taken from dictionary created in Question 6) and saves it in a file named "`Huffman_ciphered`" **(15 marks)**
2. Opens "Huffman_ciphered" and displays it **(5 marks)**
3. Calculates and displays the theoretical compression ratio (CR) in the GUI (read explanation in next page). **(5 marks)**

When a user clicks on the button "Decrypt", the program

4. Reads the ciphered file (digital), converts it into a clear text (uppercase), and saves it in a file named "Huffman_decoded" **(20 marks)**
5. Opens "Huffman_coded" and displays it **(5 marks)**

Explanation on Compression Ratio

- In computer system, information are coded by using ASCII code (see next page)
- Here, we will be using 8 bits ASCII table.
- For the compression rate, we can use the following formulas:

$$\boxed{Compression\ \ Rate = 100 - \left[\left(\frac{Size\ of\ Ciphered\ Text}{Size\ of\ Clear\ Text}\right) X\ 100\right]}$$

- Size of Ciphered Text = Text after Encryption (sum of bits form Huffman Code)
- Size of Clear Text = Text before Encryption (sum of bits from ASCII Code = sum of letters x 8 bits)

Example:

- Let's consider the following string:"centennial college"
- This will be coded as:

  By using ASCII code (since each character is 8-bits coded)

  8 bits x (10 + 1 + 7) = 144 bits

  By using Huffman Code (since the length of character is variable in HUFFMAN's table)

  (5 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 4 + 5) + (3) + (5 + 4 + 5 + 5 + 4 + 6 + 4)

  = 42 + 3 + 33

  = 78 bits

- Compression Rate = 100 – [(78/144)*100] = 45.83 %

Example of ASCII Values

	chr(32)	00100000
a	chr(97)	01100001
b	chr(98)	01100010
c	chr(99)	01100011
d	chr(100)	01100100
e	chr(101)	01100101
f	chr(102)	01100110
g	chr(103)	01100111
h	chr(104)	01101000
i	chr(105)	01101001
j	chr(106)	01101010
k	chr(107)	01101011
l	chr(108)	01101100

m	chr(109)	01101101
n	chr(110)	01101110
o	chr(111)	01101111
p	chr(112)	01110000
q	chr(113)	01110001
r	chr(114)	01110010
s	chr(115)	01110011
t	chr(116)	01110100
u	chr(117)	01110101
v	chr(118)	01110110
w	chr(119)	01110111
x	chr(120)	01111000
y	chr(121)	01111001
z	chr(122)	01111010

Example of Huffman coding from my text

(This is just an example, do not use this, yours should be populated by your text) Use the empty one provided and populate it

Letter	Frequency	Coding
Space		100
E	Most used letter	0010
T		0011
A		1111
O		1110
N		1100
I		1011
S		1010
R		0110
H		0101
L		11011
D		01111
C		01001

Letter	Frequency	Coding
U		01000
P		00011
F		00010
M		00001
W		00000
Y		110101
B		011101
G		011100
V		1101001
K		110100011
Q		110100001
X		110100000
J		1101000101
Z	Less used letter	1101000100