CSC3100 Assignment 3

Important Notes:

1. The assignment is an individual project, to be finished on one's own effort.
2. **The normal submission deadline is 6pm Apr. 10, 2023.**
3. **The late-/re-submission deadline is 6pm Apr. 13, 2023.**
4. Please also submit your final **program** and rename them as "StudentID_A2_ HuffmanCompression.java" on the blackboard. For example, a student whose Student ID is "120000001" should submit **one program** named as "120000001_A2_ HuffmanCompression.java". **You don't need to consider the consistency of class name and file name** since we won't run the code you submitted on the blackboard. File misnaming or no submission on the blackboard will lead to **5 demerit points.**
5. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism.
6. OJ website: http://oj.cuhk.edu.cn/. If you are off campus, please use VPN to access the OJ.
7. Access code: csc3100as3
8. Each student is only permitted to submit code to OJ **up to Twenty times for the problem**. Only the **last submission** will be used in evaluation of assignment marks.
9. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% of marks will be deducted.
10.

Marking Criterion:

1. For normal submission, the full score is 100 marks.
2. For late-/re-submission, the maximum score is 80 marks.
3. Four unseen test cases are used in the marking. An assignment gets 40% of full score for passing 1 case; 60% for passing 2 cases; 80% for passing 3 cases; 100% for passing all cases.

Running Environment:

1. The submissions will be evaluated in Java environment under Linux platform.
2. The submission is acceptable if it runs in any of recent versions of Java SDK environment. These versions include from Java SE 8 to the most recent Java SE 17.
3. The submission is only allowed to import three packages of (java.lang.*; java.util.*; java.io.*) included in Java SDK. No other packages are allowed.
4. In the test, each program is required to finish within **30 seconds of time**, with no more than **128MB memory**. **This is a strict requirement measured in the server environment!**
5. All students will have an opportunity to test their programs in the OJ platform prior to the official submission.

Submission Guidelines:

1. Inconsistency with or violation from the guideline leads to marks deduction.
2. It is the students' responsibility to read this assignment document and submission guidelines carefully and in detail. No argument will be accepted on issues that have been specified clearly in these documents.
3. Functional Requirement:

This assignment implements the Huffman tree (https://en.wikipedia.org/wiki/Huffman_coding), which is a powerful algorithm widely in data compression.

HuffmanCompression.java reads **one line of string** input (encoded with ASCII code) from console and then compresses the input using Huffman tree method. The program will output the compressed result as a string of "0"s and "1"s, followed by the output of the encoding dictionary..

An example of input:

```
SUSIE SAYS IT IS EASY,
```

You need to output the compression code **in a line** first and then print one possible encoding dictionary line by line as follows:

The corresponding compression output (as a string of "0"s and "1"s) will be **(only one line)**:

```
10011111011011110010010111010001100110001101000111101010111001110
```

After constructing a Huffman tree, one possible encoding dictionary (**multiple lines**) will be:

```
32:00
44:01110
65:010
69:1111
73:110
83:10
84:0110
85:01111
89:1110
```

The dictionary shows that the space character will encoded as "00" (note that 32 is the ASCII code of the space character. 44 is the ASCII code of the comma character…)

Note:

1. The input string consists of ASCII letters with possible values from **32 to 127**.
2. ASCII letters are case-sensitive. That is, "A" and "a" are different.
3. The maximum length of the input string is one million.

4. We provide 3 example test cases for you in the OJ format:

Example input 1:

```
SUSIE SAYS IT IS EASY,
```

The corresponding output could be:

```
1000011011001011110001011110111111000001111101011101000110011101 10
32:111
44:0110
65:001
69:010
73:110
83:10
84:0000
85:0001
89:0111
```

Example input 2:

```
$T_ooajX<l>=>KV8@N_r
```

The corresponding output could be:

```
0111001100011111111111001100001100010110101110101011101011010101001100011011000
1001
36:0111
56:0100
60:0010
61:1010
62:1110
64:11000
75:1011
78:11011
84:0011
86:0101
88:0110
95:000
97:11001
106:1000
108:11010
111:1111
114:1001
```

Example input 3:

```
Due to large data volume and low latency requirements of modern web services, the use of in-memory key-value (KV) cache often becomes an inevitable choice (e.g. Redis and Memcached).
```

The corresponding output could be:

```
0011000011001011101111101011101000101110110100011110111010010011111111101111100
0001010110001011001110110111001110010100101101000101010000011101000101111111111
0100100100100110110011011010001011011001110001101101111011010010111111100001100
1011111001101110101011001010101101001011000000110100111111010000101011010000011
1100010010110000000110111011111111101101110011001000010111001011111001101110000
0100011011110110111101010101101100110110001110110110011000110100001011110000101
1001011100001000011001000110000000011001000111010011110110111001011111001111111
0100101100011111010100010111101101100001100111001011011100001010100001111001111
1101110011111000110111001001111010101111000100101110000100101100111000111100111
1110001110010110010111001000011001110010100101100001010101111010100011101001111
101101100100000000100111
32:110
40:000100
41:000000
44:0001101
45:001101
46:100111
68:0011000
75:0011001
77:0001010
82:0011100
86:0001100
97:0111
98:001111
99:0100
100:10010
101:101
102:111100
103:000111
104:111101
105:11100
107:0011101
108:10001
109:11101
110:0010
111:0101
113:0001011
114:01101
115:10000
116:11111
117:01100
118:00001
119:000001
121:100110
```