Assignment 2 [6 marks]

Message Encoding/Decoding Scheme

Due: **12 Nov. 2021, 23:59** [Week 11]

1. This assignment contains **ONE** question with **Two** parts. You are required to **submit** a complete C++ program (.cpp only) **for each part** on **PASS** before the deadline.
2. You can submit as many times as you want before the deadline, and we will grade your latest version.
3. Only a small set of the test cases are visible for the testing, and a more comprehensive set of test cases will be used for grading. In other word, passing all the visible test cases does not mean that you can get full mark for this assignment. Try your best to thoroughly test your program. Hidden test cases will not be released.
4. The marking of each question is based on the percentage of the total test cases that your solution can pass. If your submitted solution leads to a compilation error on PASS, zero mark will be given to that question and no manual checking to your solution is provided in such a case.
5. No late submission will be accepted. ALL submission should be on **PASS**.
6. Plagiarism check will be performed.
7. You should **ONLY** use the library <iostream> and <cstring>. You are **NOT ALLOWED** to use objects from <string>.

In this assignment, you will write a C++ program that can decode a message under the following ***encoding scheme***.

There are two parts in this encoding scheme. The first part is a character set (header) that covers all possible characters of the messages to be decoded. The second part is an encoded message to be decoded based on the keys in the encoding scheme of the first part.

Part A. [2 marks] Part 1 Header Encoding

Write a program that reads a line of *printable* characters and encodes with a sequence of keys with each **key** is a binary ***string*** of '0's and '1's. The sequence of keys starts with a key with length 1, followed by three keys with length 2, seven keys of length 3, fifteen keys of length 4, etc. The keys of the same length are in sequence of its binary value but not with all '**1**'s.

The encoding scheme is as follows.

Assume the input header is: **n(X+# $90\"?** ...

	n	(	X	+	#		$	9	0	\	"	?	…
key string:	0	00	01	10	000	001	010	011	100	101	110	0000	…

To verify the correctness of the encoding, write a function to read in a character and print the key(s) of that character.

Notes:

1. You may not know the length of the header in advance.
2. The maximum length of the key string is seven. That is, the longest key string is "**1111110**". *Hint: A character string should have a '\0' at the end of the string.*
3. The header contains only printable characters including "*space*", i.e. any characters in the ASCII Table with values from 32 to 126.
4. We assume the input is valid, i.e., no need to check the correctness of input.
5. The header may have repeated characters that lead to different keys.

Sample Input and Output

Example 1

```
Enter Header:
n(X+# $90\"?
Character?$
010
```

Example 2

```
Enter Header:
n(X+# $90\"?#
Character?#
000
0001
```

Example 3

```
Enter Header:
n(X+# $90\"?
Character?_
001
```

input is a "space"

Part B. [4 marks] Part 2 Message Decoding

Extend your program in **Part A** to decode a message based on the keys generated in Part A.

The encoded message is a sequence of binary digits. The message should be decoded in segments. Each segment starts with the first 3 digits to represent the length of keys to be decoded in the subsequent digits. The end of the segment is signified by a sequence of '1's of that length.

> With the example header in Part A, if the segment is 010010011, the segment would be decoded as follows.

Segment

010	01	00	11
length of key	decoded to be	decoded to be	end of segment
2	X	(	

The message may have multiple segments and ends with the binary string "000".

Your program should read in the header and an encoded message; then decode the message and output the decoded message.

Notes:

1. Refer to the Notes of **Part A**.
2. The encoded message is assumed to be in one line.
3. We assume the input is valid.

Sample Input and Output

Example 1

```
Enter Header:
n(X+# $90\"?
Enter code:01100000101111100101011110111010011100 0
# 9n"X
```

Example 2

```
Enter Header:
:-+ lkC
Enter code:011010111010010111000
C++
```

Example 3

```
Enter Header:
=>?@A pqrstuv !"#$EFSTCabgh673rs-./01cde92ieo83r
Enter code:
10010111001111110101111101000101001010111110110011111011000011111011101111100001011100111111011010111011111110011001111011110111101111000010111110101011100101111110000001111101101011111101110111110110001111111000011111000
CS2311 is a great course!
```