CSE 3318 Lab Assignment 4

Due November 23

Goals:

1. Understanding of red-black trees.
2. Understanding of recursive binary tree traversal.

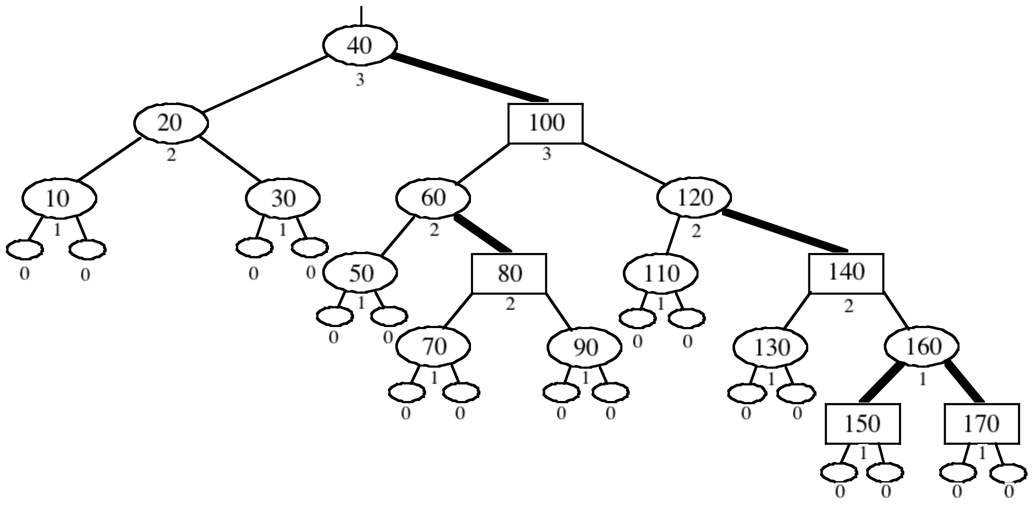
Requirements:

1. Use C to implement 1) serialization/marshalling/unloading/flattening of a red/black tree to a string and 2) the inverse operation of deserializing/unmarshalling/loading/unflattening a string to a red/black tree. Both operations are based on the recursive pre-order traversal of a binary tree.

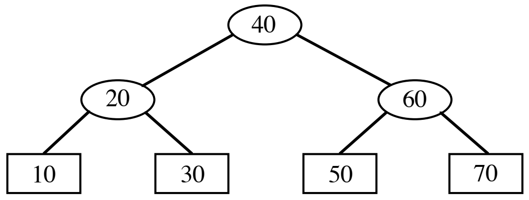
   The input is 1) the number of bytes in a string (including the NULL terminator), 2) a string no longer than the indicated length corresponding to a red-black tree, 3) `n`, the number of keys to be inserted into the tree, and 4) the `n` integers to be inserted into the tree.

   The output is 1) the length (including the NULL terminator) of a string corresponding to the final red-black tree (after insertions) and 2) the string corresponding to the final red-black tree.

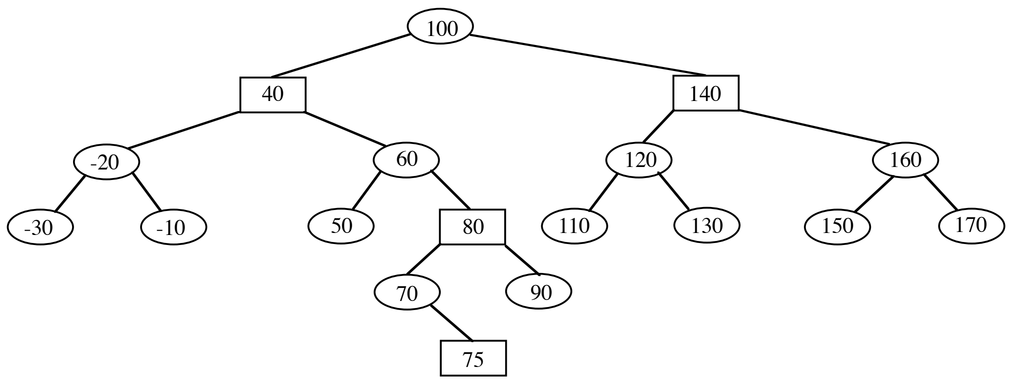
   In the serialized version of a tree, `.` indicates the sentinel. Each key will be immediately followed by a letter `r` or `b` indicating its color. Optionally, a key may include a sign (`+` or `-`). Three examples follow:



```
78 40b20b10b..30b..100r60b50b..80r70b..90b..120b110b..140r130b..160b150r..170r..
```



```
30 40b20b10r..30r..60b50r..70r..
```



```
86 100b40r-20b-30b..-10b..60b50b..80r70b.75r..90b..140r120b110b..130b..+160b150b..170b..
```

2. Submit all necessary C source files on Canvas by 5:00 pm on November 23. Comments at the beginning of the source file should include: your name, your ID number, and the command used to compile your code on Omega (5 point penalty for non-compliance).

Getting Started:

1. Suitable driver and header files are available at `http://ranger.uta.edu/~weems/NOTES3318/LAB/LAB4FALL21/` . `RB.c` and its `RB.h` header file are available at `http://ranger.uta.edu/~weems/NOTES3318/REDBLACKC/` .

2. You must use separate compilation. Do not merge together implementation and header files.

3. The string representing a red-black tree will be free of spaces.

4. Be sure your code does not leak memory. If you `malloc()` it, you are obligated to `free()` it.

5. You should check the deserialized tree either while building it or by using `verifyRBproperties()`.

6. Your deserialization code should check the input string for errors. Characters past the end of a tree should result in a warning. Inappropriate characters elsewhere should result in a message and `exit()` termination.

7. `sprintf()` in `stdio.h` will be very useful for string concatenation in your serialization code.