Cover Page

Name: David Chen Salas

Section: 2023 Fall Term (1) Algorithms I CSCI 700 231[25504] (Queens College)

Project#: 3

Project Name: Huffman coding Part 2

Due Date: 10/2/2023 Monday before midnight

Algorithm Steps:

```
Step 0: InFile, outFile1, deBugFile , deBugFile2  open via args [0], args [1], args [2], args [3]
Step 1: computeCharCounts (inFile, charCountAry, deBugFile) // On your own,
        // You may pass deBugFile to write some debugging prints
Step 2: printCountAry (charCountAry, outFile1) // with caption "Below is character counts"
Step 3: LL  creates a LLlist using constructor and assign
        LL.listHead  get a treeNode with ("dummy" , 0, '', null, null, null) // '' is an empty string
Step 4: constructHuffmanLList (LL.listHead, charCountAry, deBugFile)
Step 5: printList (LL.listHead, outFile1) // with caption "Below is the ordered Huffman ordered
Linked list."
Step 6: constructHuffmanBinTree (LL.listHead, deBugFile)
Step 7: (binTree) HuffmanTree  create a binTree node using binTree constructor and assign
Step 8: HuffmanTree.Root  LL.listHead.next
Step 9: preOrder (HuffmanTree.Root, deBugFile)
        // with caption "Below is preOrder of the Huffman Binary Tree"
        inOrder (HuffmanTree.Root, deBugFile) // with caption.
        postOrder (HuffmanTree.Root, deBugFile) // with caption.
Step 10: constructCharCode (HuffmanTree.Root, '', codeTable) // '' is an empty string.
**** part 2 start here ***
Step 11: printCodeTable (codeTable, outFile1)
Step 12: userInterface (HuffmanTree.Root, codeTable, deBugFile2)
Step 13: close all files.
```

Source Code

```java
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import java.lang.String;

public class ChenSalasD_Project3_Main {

    static Scanner inFile;
    static FileWriter outFile1;
    static FileWriter deBugFile, deBugFile2;
    static HuffmanCode huffmanCode;
    static LLlist LL;
    static binTree huffmanTree;
```

```
public static void main(String[] args) throws IOException {

    inFile = new Scanner(new FileReader(args[0]));
    inFile.useDelimiter("");
    outFile1 = new FileWriter(args[1]);
    deBugFile = new FileWriter(args[2]);
    deBugFile2 = new FileWriter(args[3]);
    huffmanCode = new HuffmanCode();

    computeCharCounts(inFile, huffmanCode.charCountAry, deBugFile);
    printCountAry(huffmanCode.charCountAry, outFile1);
    LL = new LLlist();
    constructHuffmanLList(LL.listHead, huffmanCode.charCountAry, deBugFile);
    printList(LL.listHead, outFile1);
    constructHuffmanBinTree(LL.listHead, deBugFile);
    huffmanTree = new binTree(LL.listHead.next);

    deBugFile.write("\n**Below is preOrder of the Huffman Binary Tree**\n");
    preOrder(huffmanTree.Root, deBugFile);
    deBugFile.write("\n**Below is inOrder of the Huffman Binary Tree**\n");
    inOrder(huffmanTree.Root, deBugFile);
    deBugFile.write("\n**Below is postOrder of the Huffman Binary Tree**\n");
    postOrder(huffmanTree.Root, deBugFile);
    constructCharCode(huffmanTree.Root, "", huffmanCode.codeTable);

    printCodeTable(huffmanCode.codeTable, outFile1);
    userInterface(huffmanTree.Root, huffmanCode.codeTable, deBugFile2);

    inFile.close();
    outFile1.close();
    deBugFile.close();
}

public static void printNode(treeNode T, FileWriter file) throws IOException {
    String chStr, leftChr, rightChr, nextChr;

    switch (T.chStr){
        case " ":
            chStr = "\'space\'";
            break;
        case "\r":
            chStr = "\\r";
            break;
        case "\n":
            chStr = "NL";
            break;
        default:
            chStr = T.chStr;
    }

    if(T.left == null) leftChr = "null";
    else {
        switch (T.left.chStr){
            case " ":
                leftChr = "\'space\'";
                break;
            case "\r":
                leftChr = "\\r";
                break;
            case "\n":
                leftChr = "NL";
                break;
            default:
                leftChr = T.left.chStr;
        }
    }

    if(T.right == null) rightChr = "null";
    else {
        switch (T.right.chStr){
            case " ":
                rightChr = "\'space\'";
                break;
            case "\r":
                rightChr = "\\r";
                break;
            case "\n":
                rightChr = "NL";
                break;
            default:
                rightChr = T.right.chStr;
        }
    }

    if(T.next == null) nextChr = "null";
    else {
        switch (T.next.chStr) {
            case " ":
                nextChr = "\'space\'";
                break;
            case "\r":
                nextChr = "\\r";
                break;
            case "\n":
                nextChr = "NL";
                break;
            default:
                nextChr = T.next.chStr;
        }
    }

    file.write("(");
    file.write(chStr+",  " + T.frequency+",  " + T.code+",  " + leftChr+",  " + rightChr+",  " + nextChr);
    file.write(")\n");
}

public static void printList(treeNode listHead, FileWriter file) throws IOException {
    if(file.toString()=="outFile1"){
        file.write("Below is the ordered Huffman ordered Linked list.\n");
    }
    else {
        file.write("Printing list in constructHuffmanLList method.\n");
    }
    treeNode pNode = listHead;
    while(pNode!=null){
        printNode(pNode, file);
        pNode = pNode.next;
    }
}

public static treeNode findSpot(treeNode listHead, treeNode newNode, FileWriter deBugFile) throws IOException {
    deBugFile.write("Entering findSpot method!\n");
    treeNode spot = listHead;
    while (spot.next != null) {
        deBugFile.write("In findSpot: Spot.next's frequency is " + spot.next.frequency +
                " and newNode's frequency is " + newNode.frequency + "\n");
        if (spot.next.frequency < newNode.frequency) {
            spot = spot.next;
        }
        else{
            break;
        }
    }
    deBugFile.write("Leaving findSpot method!\n");
    return spot;
}

public static void insertOneNode(treeNode spot, treeNode newNode){
    newNode.next = spot.next;
    spot.next = newNode;
}

public static void preOrder(treeNode rootNode, FileWriter file) throws IOException {
    if(rootNode.left==null && rootNode.right==null){
        printNode(rootNode, file);
    }
    else {
        printNode(rootNode,file);
        preOrder(rootNode.left, file);
        preOrder(rootNode.right, file);
    }
}

public static void inOrder(treeNode rootNode, FileWriter file) throws IOException{

    if(rootNode.left==null && rootNode.right==null){
        printNode(rootNode, file);
    }
    else {
        inOrder(rootNode.left, file);
        printNode(rootNode,file);
        inOrder(rootNode.right,file);
    }
}

public static void postOrder(treeNode rootNode, FileWriter file) throws IOException {

    if(rootNode.left==null && rootNode.right==null){
        printNode(rootNode, file);
    }
    else {
        postOrder(rootNode.left, file);
        postOrder(rootNode.right, file);
        printNode(rootNode,file);
    }
}

public static boolean isLeaf(treeNode node){
    if(node.left==null && node.right==null){
        return true;
    }
    return false;
}

public static void computeCharCounts(Scanner inFile, int[] charCountAry, FileWriter deBugFile) throws IOException {
        deBugFile.write("Entering computeCharCounts method!\n");
        char c;
        while(inFile.hasNext()) {
            c = (char)inFile.next().charAt(0);
            charCountAry[(int)c]++;
        }
        deBugFile.write("Leaving computeCharCounts method!\n");
    }

    public static void printCountAry(int[] charCountAry, FileWriter outFile1) throws IOException {
        outFile1.write("**Below is character counts**\n");
        outFile1.write("Index\tChar\tCount\n");
        outFile1.write("====================\n");
        for(int i=0; i<charCountAry.length; i++){
            if(charCountAry[i]!=0) {
                if(i==10){
                    outFile1.write(i + "\tNL\t" + charCountAry[i] + "\n");
                }
                else if(i==13){
                    outFile1.write(i + "\t\\r\t" + charCountAry[i] + "\n");
                }
                else if(i==32){
                    outFile1.write(i + "\t'space'\t" + charCountAry[i] + "\n");
                }
                else{
                    outFile1.write(i + "\t" + (char)i + "\t" + charCountAry[i] + "\n");
                }
            }
        }
    }

    public static void constructHuffmanLList(treeNode listHead, int[] charCountAry, FileWriter deBugFile) throws IOException
{
        deBugFile.write("Entering constructHuffmanLList method!\n");
        char chr;
        int frequency;
        for(int i=0; i<256; i++){
            if(charCountAry[i]>0){
                chr = (char)i;
                frequency = charCountAry[i];
                treeNode newNode = new treeNode(""+chr, frequency,"",null, null, null);
                printNode(newNode, deBugFile);
                treeNode spot = findSpot(listHead, newNode, deBugFile);
                insertOneNode(spot, newNode);
                printList(listHead, deBugFile);
            }
        }
        deBugFile.write("Leaving constructHuffmanLList method!\n");
    }

    public static void constructHuffmanBinTree(treeNode listHead, FileWriter deBugFile) throws IOException{
        deBugFile.write("Entering constructHuffmanBinTree method!\n");
        while(listHead.next.next!=null) {
            treeNode leftNode = listHead.next;
            treeNode rightNode = listHead.next.next;
            String cStr = leftNode.chStr + rightNode.chStr;
            StringBuilder s = new StringBuilder();
            for(char x: cStr.toCharArray()){
                if(x == ' ') s.append("\'space\'");
                else if(x == '\r') s.append("\\r");
                else if(x == '\n') s.append("NL");
                else s.append(x);
            }
            int frequency = leftNode.frequency + rightNode.frequency;
            treeNode newNode = new treeNode(s.toString(), frequency, "", leftNode, rightNode, null);
            printNode(newNode, deBugFile);
            treeNode spot = findSpot(listHead, newNode, deBugFile);
            insertOneNode(spot, newNode);
            listHead.next = listHead.next.next.next;
            printList(listHead, deBugFile);
        }
        deBugFile.write("Leaving constructHuffmanBinTree method!\n");
    }

    public static void constructCharCode(treeNode T, String code, String[] codeTable){
        if(isLeaf(T)){
            T.code = code;
            codeTable[(int)T.chStr.charAt(0)] = code;
        }
        else{
            constructCharCode(T.left, code+"0", codeTable);
            constructCharCode(T.right, code+"1", codeTable);
        }
    }

    public static void printCodeTable(String codeTable[], FileWriter outFile1) throws IOException {
        outFile1.write("**Below is code table**\n");
        outFile1.write("Index\tChar\tCount\n");
        outFile1.write("=====================\n");
        for(int i=0; i<codeTable.length; i++){
            if(codeTable[i] != null) {
                if(i==10){
                    outFile1.write(i + "\tNL\t" + codeTable[i] + "\n");
                }
                else if(i==13){
                    outFile1.write(i + "\t\\r\t" + codeTable[i] + "\n");
                }
                else if(i==32){
                    outFile1.write(i + "\t'space'\t" + codeTable[i] + "\n");
                }
                else{
                    outFile1.write(i + "\t" + (char)i + "\t" + codeTable[i] + "\n");
                }
            }
        }
    }

    public static void userInterface(treeNode Root, String codeTable[], FileWriter deBugFile2) throws IOException {
        String nameOrg, nameCompress, nameDeCompress;
        char yesNo;
        while (true) {
            System.out.print("Encode a file? (Y or any to encode, and N to exit program): ");
            yesNo = System.console().readLine().charAt(0);
            if (yesNo == 'N') {
                return;
            }
            System.out.print("Name of file want to encode: ");
            nameOrg = System.console().readLine();
            nameCompress = nameOrg + "_Compressed.txt";
            nameDeCompress = nameOrg + "_deCompressed.txt";
            nameOrg = nameOrg + ".txt";

            Scanner orgFile = new Scanner(new FileReader(nameOrg));
            orgFile.useDelimiter("");
```

```
            FileWriter compFile = new FileWriter(nameCompress);
            FileWriter deCompFile = new FileWriter(nameDeCompress);

            Encode(orgFile, compFile, codeTable, deBugFile2);
            compFile.close();
            Scanner rCompFile = new Scanner(new FileReader(nameCompress));
            rCompFile.useDelimiter("");
            Decode(rCompFile, deCompFile, Root, deBugFile2);

            orgFile.close();
            compFile.close();
            deCompFile.close();

        }
    }

    public static void Encode(Scanner FileIn, FileWriter FileOut, String codeTable[], FileWriter deBugFile2) throws
IOException{
        deBugFile2.write("Entering Encode method!\n");
        char charIn;
        int index;
        String code;
        while(FileIn.hasNext()){
            charIn = FileIn.next().charAt(0);
            index = (int)charIn;
            code = codeTable[index];
            FileOut.write(code);
            deBugFile2.write("inside Encode(): index="+ index +" code=" + code + "\n");
        }
        deBugFile2.write("Leaving Encode method!\n");
    }

    public static void Decode(Scanner FileIn, FileWriter FileOut, treeNode Root, FileWriter deBugFile2) throws IOException{
        deBugFile2.write("Entering Decode method!\n");
        treeNode spot = Root;
        char oneBit;
        while(FileIn.hasNext()){
            if(isLeaf(spot)){
                FileOut.write(spot.chStr);
                deBugFile2.write("Inside Decode method: " + spot.chStr + "\n");
                spot = Root;
            }
            oneBit = FileIn.next().charAt(0);
            if(oneBit == '0'){
                spot = spot.left;
            }
            else if(oneBit == '1'){
                spot = spot.right;
            }
            else{
                deBugFile2.write("Error! The compress file contains invalid character!\n");
                return;
            }
        }
        if(!isLeaf(spot)){
            deBugFile2.write("Error: The compress file is corrupted!\n");
            return;
        }
        deBugFile2.write("Leaving Decode method!\n");
    }

}
```

```
class treeNode{
    String chStr;
    int frequency;
    String code;
    treeNode left;
    treeNode right;
    treeNode next;

    treeNode(String chStr, int frequency, String code, treeNode left,
             treeNode right, treeNode next){
        this.chStr = chStr;
        this.frequency = frequency;
        this.code = code;
        this.left = left;
        this.right = right;
        this.next = next;
    }
}

class LLlist{
    treeNode listHead;
    LLlist(){
        listHead = new treeNode("dummy", 0, "", null, null, null);
    }
}

class binTree{
    treeNode Root;
    binTree(treeNode Root){
        this.Root = Root;
    }
}

class HuffmanCode {
    int[] charCountAry;
    String[] codeTable;

    HuffmanCode() {
        charCountAry = new int[256];
        codeTable = new String[256];
    }
}
```

Program Output

Below is outFile1.txt

```
*Below is character counts*
Index   Char    Count
====================
10      NL      399
13      \r      399
32      'space' 4253
39      '       15
40      (       1
41      )       1
44      ,       373
45      -       1
46      .       181
49      1       23
51      3       11
52      4       1
53      5       2
54      6       11
56      8       13
57      9       11
59      ;       1
65      A       47
66      B       23
67      C       34
68      D       1
70      F       12
71      G       46
72      H       3
73      I       44
74      J       1
76      L       22
77      M       6
78      N       33
79      O       6
80      P       23
83      S       40
84      T       49
85      U       25
87      W       34
97      a       1627
98      b       271
99      c       456
100     d       876
101     e       2547
102     f       478
103     g       406
104     h       1240
105     i       1131
106     j       1
107     k       81
108     l       662
109     m       269
110     n       1321
111     o       1436
112     p       216
113     q       11
114     r       1226
115     s       862
```

```
116	t	1914
117	u	343
118	v	323
119	w	338
120	x	6
121	y	260
Printing list in constructHuffmanLList method.
(dummy,  0,  ,  null,  null,  j)
(j,  1,  ,  null,  null,  J)
(J,  1,  ,  null,  null,  D)
(D,  1,  ,  null,  null,  ;)
(;,  1,  ,  null,  null,  4)
(4,  1,  ,  null,  null,  -)
(-,  1,  ,  null,  null,  ))
(),  1,  ,  null,  null,  ()
((,  1,  ,  null,  null,  5)
(5,  2,  ,  null,  null,  H)
(H,  3,  ,  null,  null,  x)
(x,  6,  ,  null,  null,  O)
(O,  6,  ,  null,  null,  M)
(M,  6,  ,  null,  null,  q)
(q,  11,  ,  null,  null,  9)
(9,  11,  ,  null,  null,  6)
(6,  11,  ,  null,  null,  3)
(3,  11,  ,  null,  null,  F)
(F,  12,  ,  null,  null,  8)
(8,  13,  ,  null,  null,  ')
(',  15,  ,  null,  null,  L)
(L,  22,  ,  null,  null,  P)
(P,  23,  ,  null,  null,  B)
(B,  23,  ,  null,  null,  1)
(1,  23,  ,  null,  null,  U)
(U,  25,  ,  null,  null,  N)
(N,  33,  ,  null,  null,  W)
(W,  34,  ,  null,  null,  C)
(C,  34,  ,  null,  null,  S)
(S,  40,  ,  null,  null,  I)
(I,  44,  ,  null,  null,  G)
(G,  46,  ,  null,  null,  A)
(A,  47,  ,  null,  null,  T)
(T,  49,  ,  null,  null,  k)
(k,  81,  ,  null,  null,  .)
(.,  181,  ,  null,  null,  p)
(p,  216,  ,  null,  null,  y)
(y,  260,  ,  null,  null,  m)
(m,  269,  ,  null,  null,  b)
(b,  271,  ,  null,  null,  v)
(v,  323,  ,  null,  null,  w)
(w,  338,  ,  null,  null,  u)
(u,  343,  ,  null,  null,  ,)
(,,  373,  ,  null,  null,  \r)
(\r,  399,  ,  null,  null,  NL)
(NL,  399,  ,  null,  null,  g)
(g,  406,  ,  null,  null,  c)
(c,  456,  ,  null,  null,  f)
(f,  478,  ,  null,  null,  l)
(l,  662,  ,  null,  null,  s)
(s,  862,  ,  null,  null,  d)
(d,  876,  ,  null,  null,  i)
(i,  1131,  ,  null,  null,  r)
(r,  1226,  ,  null,  null,  h)
(h,  1240,  ,  null,  null,  n)
```

```
(n,  1321,  ,  null,  null,  o)
(o,  1436,  ,  null,  null,  a)
(a,  1627,  ,  null,  null,  t)
(t,  1914,  ,  null,  null,  e)
(e,  2547,  ,  null,  null,  'space')
('space',  4253,  ,  null,  null,  null)
**Below is code table**
Index   Char     Count
====================
10      NL       100110
13      \r       100101
32      'space'  111
39      '        0001010011
40      (        01111100000101
41      )        01111100000100
44      ,        100100
45      -        01111100000111
46      .        0111111
49      1        1001110010
51      3        01111100011
52      4        01111100000110
53      5        0111110000100
54      6        01111100010
56      8        0001010010
57      9        01111100101
59      ;        01111100000001
65      A        100111010
66      B        0111110111
67      C        000101100
68      D        01111100000000
70      F        10011100111
71      G        100111000
72      H        0111110000101
73      I        011111010
74      J        01111100000011
76      L        0111110011
77      M        100111001101
78      N        000101010
79      O        100111001100
80      P        0111110110
83      S        000101101
84      T        100111011
85      U        000101000
87      W        000101011
97      a        1010
98      b        000100
99      c        101101
100     d        11000
101     e        010
102     f        110010
103     g        101100
104     h        0011
105     i        0000
106     j        01111100000010
107     k        00010111
108     l        01110
109     m        1100111
110     n        0110
111     o        1000
112     p        1001111
113     q        01111100100
114     r        0010
```

115	s	10111
116	t	1101
117	u	011110
118	v	000110
119	w	000111
120	x	011111000011
121	y	1100110

Below is test1

```
The boy visits Santiago's shack each night,
hauling his fishing gear, preparing food,
talking about American baseball and his favorite player,
Joe DiMaggio. Santiago tells Manolin that on the next day,
he will venture far out into the Gulf Stream,
north of Cuba in the Straits of Florida to fish,
confident that his unlucky streak is near its end.
```

Below is output test1_compressed

```
1001110110011010111000100100011001101110001100000101110000110110111111000101101
1010011011010000101010110010000001010011101111111011100111010101101000101111110
1010101011010011111011000001011000011110110010011110010110011000111010011110011
1000000110101100111001100001011111111001000001011100110000011010110011110110001
0101000101001001111001111001001010011111010001000000110101100111110010100010001
1000100100111100101100110110110100111000010111000001101011001111010000100100001
1110110111110011101011001110100010000010110110100110111000100101010111010000100
1010011100111011110100110110001110011000010111111110010101000011010000010000011
0101011110011110111010101100110010001010010011110010110011001111100000011100001
011101111100000000000000100111001101101010110010110000001000011111111100010110110
1001101101000010101011001000111110101001110011101011111110011100110110100110100
0011100000011011111010011101011011111000011011111010011010111011001001111000011
1101111110001010110011010010011110010110011000110101110001110000011100111011100
0110010011011010111100010010111110010101000101111000011110110111100000110110110
0011111010011010111100111000011110011101100101110001011011101001001010101100111
1001001111001011001100110100000101101001111110001100101110001011000111100001001
0101110000011011111010011010111000101101110100101010000011011011111110001100101
11100111001110111010000001000001100010101111101100011111001000001011100111001001
1110010110011010110110000110110010000011000010011011011111101001110101101111001
10000101111110111100110011100111101011010001011111001101111011111010010010101000
0101111110000101111110110010101000101110000110110111111010011011000011111111
```

Below is output test1_deCompressed

```
The boy visits Santiago's shack each night,
hauling his fishing gear, preparing food,
talking about American baseball and his favorite player,
Joe DiMaggio. Santiago tells Manolin that on the next day,
he will venture far out into the Gulf Stream,
north of Cuba in the Straits of Florida to fish,
confident that his unlucky streak is near its end.
```

****Below is test2****

The Gettysburg Address is a speech by U.S. President Abraham Lincoln,
one of the best known in American history.
It was delivered by Lincoln during the American Civil War,
on the afternoon of Thursday, November 19, 1863, at the dedication of the Soldiers'
National Cemetery in Gettysburg, Pennsylvania,
four and a half months after the Union armies defeated those of the Confederacy
at the Battle of Gettysburg.

Below is output test2_compressed

```
10011101100110101111001110000101101110111001101011100010001111000101011001111001
11101011000110000010010101111011111100001011111110101111011110011110100101011010
01111100010011001101110001010000111111000101101011111111101111011000100101011100
00011000010011011011111001110100001000010101000111010110011111101111110011000001
10101101100001110011010010011110010110011010000110010111100011001011111010011010
01110001000101011111011110001011101101000000111011011100000110111100111010110011
11010001000000101101101001101110011000010111110110000010110011001111111111001011
00110011111010110111100011110101011111111000010011100000000110010001001011000111
00010011001101110111110011000000110101101100001110011011111000011110001000000110
10110011111010011010111100111010110011101000100000101101101001101110001011000000
00001100000001110111000101011101000101001001111001011001101000011011111010011010
01111010110010110101000100110100010000110111100011001011110011101100110111100010
10111110001010110011010010011100010101010000000110010110011100010001000101111001
11100100111110010110010011110011100100001010010011111000100111110001110010011110
10110111111010011010111110000101100000000101101101011010000100001101111000110010
01111101001101011100010110110000111011000000000100010101110001010011111100101100
11000010101010101101000010000110101001110111000101100010110011101011010100010110
01101110000011011110011100001011011101110011010111000100011110001010110010010011
10111111011001001100110101111100110011100001101010011000001010100100111100101100
11011001010000111100010111101001101100011110101110011101001110110010111110011111
00001101101001110111111101011001011010100010111110100110101110001010000110000010
00001101111010001011001110000010101111111100001011001001010101101010110001111110
10011100010111010111100011001011111010011010111000101100100001101100100101100001
00010101010110111001101111001011001101010110111111010011010111011111011110101101
11010111001011110001100101111001110000101101110111001101011100010001111000101011
000111111111
```

Below is output test2_deCompressed

The Gettysburg Address is a speech by U.S. President Abraham Lincoln,
one of the best known in American history.
It was delivered by Lincoln during the American Civil War,
on the afternoon of Thursday, November 19, 1863, at the dedication of the Soldiers'
National Cemetery in Gettysburg, Pennsylvania,
four and a half months after the Union armies defeated those of the Confederacy
at the Battle of Gettysburg.

****Below is test3****
Santiago straps the marlin to the side of his skiff and heads home,
thinking about the high price the fish will bring him at the market
and how many people he will feed.
On his way in to shore, sharks are attracted to the marlin's blood.
Santiago kills a great mako shark with his harpoon,
but he loses the weapon.
He makes a new harpoon by strapping his knife
to the end of an oar to help ward off the next line of sharks;
five sharks are slain and many others are driven away.
But the sharks keep coming, and by nightfall
the sharks have almost devoured the marlin's entire carcass,
leaving a skeleton consisting mostly of its backbone, its tail,
and its head. Santiago knows that he is destroyed and
tells the sharks of how they have killed his dreams.
Below is output test3_compressed

```
00010110110100110110100001010101100100011110111110100101010100111110111111111010
01101011111001111010001001110000001101111101100011111010011010111101110000110000
10111100011001011100110000101111111011100010111000011001011001011110100110110000
11100110101010110001011111100111000110011101010010011110010110011011010011000000
11000010111000000110101100111101000010010000111101101111110100110101110011000010
11000011111100111100100000101101010111110100110101111100100000101110011111000111
00000111001110111000100001000000110101100111001100001100111111101011011111110100
11010111110011110100010000101110101101111100101100110101001101100011100111000000
01111111100111101001101100110111100111101010001001111011100101110011010111000111
00000111001110111110010010010110000111111111100101100110100111001100011011100110
00010111111000111101011001101110000011011111011000111101110011100000010010100100
11110111001110100010000101111011111110100010010111101011011101001010101011011110
10101100011111011000111110100110101111100111101000100111000000110000101001110111
11110001000111010001000110000111111111001011001100001011011010011011010000101010
10110010001110001011100000111001110101111111010111101100001001010101101111110011
11101000010111100011110111001110100010000101111110001110000110100111110011000001
01111110011101000101001111100010000110100100111100101100110000100011110110111110
01101011101110100010111010101111111101001101011100011101010101001111100001100111
11111110010110011001111100001010101111100111101000010111010101111111010111011000
10000111111001110100010100111110001000011011100010011001101111011111010010101010
00111111001111000000110101100111001100001011111100010111011000000110010010111100
10110011011011000111110100110101110100110110001111000110010111101001101111000101
00010111110110001110011010011101001111111000111101000101100011110001100101100101
11110100110101110110010011111000011110111101110000000110010111100011001011110111
00111010001000010111101110111110000000111110010110011011001000000000110010111101
11001110100010000101111011111110100010010111101110111010100000011011110100110110
00111110011110100110110011011110001101001101000101011111110100010010111110000010
00000000110010011011110100001111010110011001111111111001011001100111101110111101
01101111110100110101111011100111010001000010111101111110001011101001010011111111
01101100011001110000011010110010010011110100110110001110001001100110111011000001
```

```
0110000111101110010101001110011101111001011001101101001101011110111001110100010
0001011110111111001110100001100101111010011101100111100010111110111111000010000 1
1010000111100010010110001111101001101011111001111010001001110000000110000101001 1
1011111101001101101000000100101111011011010001010110110101011110111100100111100
1011001100111001010100001100000011010110011110101111011100010111010011100101101
1000011011110110110000110101110000101111101000001101011001111100111100010111110
1011101100110111100011001011100001101101111110001001010101101000101110001001000
0110010100100111000011011011111111011010000001110100100111100101100110101001101
1000111000011011011111100110101010110000111111111000101101101001101101000010101
0110010001110001011101101000000111101111111101001110101101111001101011100001011 1
1111100001010111110100101000110011001011000111101001101100011110010110011011010
1001110011101011111111010011010111101110011101000100001011110111111100011001011 1
0011100000011111111010011010110011011100111010000110010111000101110000011100111
00101100011100110000101111111100000100101010110011110111011111111
```
Below is output test3_deCompressed
Santiago straps the marlin to the side of his skiff and heads home,
thinking about the high price the fish will bring him at the market
and how many people he will feed.
On his way in to shore, sharks are attracted to the marlin's blood.
Santiago kills a great mako shark with his harpoon,
but he loses the weapon.
He makes a new harpoon by strapping his knife
to the end of an oar to help ward off the next line of sharks;
five sharks are slain and many others are driven away.
But the sharks keep coming, and by nightfall
the sharks have almost devoured the marlin's entire carcass,
leaving a skeleton consisting mostly of its backbone, its tail,
and its head. Santiago knows that he is destroyed and
tells the sharks of how they have killed his dreams.

****Below is deBugFile2****

```
Entering Encode method!
inside Encode(): index=84 code=100111011
inside Encode(): index=104 code=0011
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=98 code=000100
inside Encode(): index=111 code=1000
inside Encode(): index=121 code=1100110
inside Encode(): index=32 code=111
inside Encode(): index=118 code=000110
inside Encode(): index=105 code=0000
inside Encode(): index=115 code=10111
inside Encode(): index=105 code=0000
inside Encode(): index=116 code=1101
inside Encode(): index=115 code=10111
inside Encode(): index=32 code=111
inside Encode(): index=83 code=000101101
inside Encode(): index=97 code=1010
inside Encode(): index=110 code=0110
inside Encode(): index=116 code=1101
inside Encode(): index=105 code=0000
inside Encode(): index=97 code=1010
inside Encode(): index=103 code=101100
inside Encode(): index=111 code=1000
inside Encode(): index=39 code=0001010011
inside Encode(): index=115 code=10111
inside Encode(): index=32 code=111
inside Encode(): index=115 code=10111
inside Encode(): index=104 code=0011
inside Encode(): index=97 code=1010
inside Encode(): index=99 code=101101
inside Encode(): index=107 code=00010111
inside Encode(): index=32 code=111
inside Encode(): index=101 code=010
inside Encode(): index=97 code=1010
inside Encode(): index=99 code=101101
inside Encode(): index=104 code=0011
inside Encode(): index=32 code=111
inside Encode(): index=110 code=0110
inside Encode(): index=105 code=0000
inside Encode(): index=103 code=101100
inside Encode(): index=104 code=0011
inside Encode(): index=116 code=1101
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=13 code=100101
inside Encode(): index=10 code=100110
inside Encode(): index=104 code=0011
inside Encode(): index=97 code=1010
inside Encode(): index=117 code=011110
inside Encode(): index=108 code=01110
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=103 code=101100
inside Encode(): index=32 code=111
inside Encode(): index=104 code=0011
inside Encode(): index=105 code=0000
inside Encode(): index=115 code=10111
inside Encode(): index=32 code=111
inside Encode(): index=102 code=110010
inside Encode(): index=105 code=0000
inside Encode(): index=115 code=10111
inside Encode(): index=104 code=0011
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=103 code=101100
inside Encode(): index=32 code=111
inside Encode(): index=103 code=101100
inside Encode(): index=101 code=010
inside Encode(): index=97 code=1010
inside Encode(): index=114 code=0010
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=112 code=1001111
inside Encode(): index=114 code=0010
inside Encode(): index=101 code=010
inside Encode(): index=112 code=1001111
inside Encode(): index=97 code=1010
inside Encode(): index=114 code=0010
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=103 code=101100
inside Encode(): index=32 code=111
inside Encode(): index=102 code=110010
inside Encode(): index=111 code=1000
inside Encode(): index=111 code=1000
inside Encode(): index=100 code=11000
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=13 code=100101
inside Encode(): index=10 code=100110
inside Encode(): index=116 code=1101
inside Encode(): index=97 code=1010
inside Encode(): index=108 code=01110
inside Encode(): index=107 code=00010111
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=103 code=101100
inside Encode(): index=32 code=111
inside Encode(): index=97 code=1010
inside Encode(): index=98 code=000100
inside Encode(): index=111 code=1000
inside Encode(): index=117 code=011110
inside Encode(): index=116 code=1101
inside Encode(): index=32 code=111
inside Encode(): index=65 code=100111010
inside Encode(): index=109 code=1100111
inside Encode(): index=101 code=010
inside Encode(): index=114 code=0010
inside Encode(): index=105 code=0000
inside Encode(): index=99 code=101101
inside Encode(): index=97 code=1010
inside Encode(): index=110 code=0110
inside Encode(): index=32 code=111
inside Encode(): index=98 code=000100
inside Encode(): index=97 code=1010
inside Encode(): index=115 code=10111
inside Encode(): index=101 code=010
inside Encode(): index=98 code=000100
inside Encode(): index=97 code=1010
inside Encode(): index=108 code=01110
inside Encode(): index=108 code=01110
inside Encode(): index=32 code=111
inside Encode(): index=97 code=1010
inside Encode(): index=110 code=0110
inside Encode(): index=100 code=11000
inside Encode(): index=32 code=111
inside Encode(): index=104 code=0011
inside Encode(): index=105 code=0000
inside Encode(): index=115 code=10111
inside Encode(): index=32 code=111
inside Encode(): index=102 code=110010
inside Encode(): index=97 code=1010
inside Encode(): index=118 code=000110
inside Encode(): index=111 code=1000
inside Encode(): index=114 code=0010
inside Encode(): index=105 code=0000
inside Encode(): index=116 code=1101
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=112 code=1001111
inside Encode(): index=108 code=01110
inside Encode(): index=97 code=1010
inside Encode(): index=121 code=1100110
inside Encode(): index=101 code=010
inside Encode(): index=114 code=0010
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=13 code=100101
inside Encode(): index=10 code=100110
inside Encode(): index=74 code=01111100000011
inside Encode(): index=111 code=1000
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=68 code=01111100000000
inside Encode(): index=105 code=0000
inside Encode(): index=77 code=100111001101
inside Encode(): index=97 code=1010
inside Encode(): index=103 code=101100
inside Encode(): index=103 code=101100
inside Encode(): index=105 code=0000
inside Encode(): index=111 code=1000
inside Encode(): index=46 code=0111111
inside Encode(): index=32 code=111
inside Encode(): index=83 code=000101101
inside Encode(): index=97 code=1010
inside Encode(): index=110 code=0110
inside Encode(): index=116 code=1101
inside Encode(): index=105 code=0000
inside Encode(): index=97 code=1010
inside Encode(): index=103 code=101100
inside Encode(): index=111 code=1000
inside Encode(): index=32 code=111
inside Encode(): index=116 code=1101
inside Encode(): index=101 code=010
inside Encode(): index=108 code=01110
inside Encode(): index=108 code=01110
inside Encode(): index=115 code=10111
inside Encode(): index=32 code=111
inside Encode(): index=77 code=100111001101
inside Encode(): index=97 code=1010
inside Encode(): index=110 code=0110
inside Encode(): index=111 code=1000
inside Encode(): index=108 code=01110
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=32 code=111
inside Encode(): index=116 code=1101
inside Encode(): index=104 code=0011
inside Encode(): index=97 code=1010
inside Encode(): index=116 code=1101
inside Encode(): index=32 code=111
inside Encode(): index=111 code=1000
inside Encode(): index=110 code=0110
inside Encode(): index=32 code=111
inside Encode(): index=116 code=1101
inside Encode(): index=104 code=0011
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=110 code=0110
inside Encode(): index=101 code=010
inside Encode(): index=120 code=011111000011
inside Encode(): index=116 code=1101
inside Encode(): index=32 code=111
inside Encode(): index=100 code=11000
inside Encode(): index=97 code=1010
inside Encode(): index=121 code=1100110
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=13 code=100101
inside Encode(): index=10 code=100110
inside Encode(): index=104 code=0011
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=119 code=000111
inside Encode(): index=105 code=0000
inside Encode(): index=108 code=01110
inside Encode(): index=108 code=01110
inside Encode(): index=32 code=111
inside Encode(): index=118 code=000110
inside Encode(): index=101 code=010
inside Encode(): index=110 code=0110
inside Encode(): index=116 code=1101
inside Encode(): index=117 code=011110
inside Encode(): index=114 code=0010
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=102 code=110010
inside Encode(): index=97 code=1010
inside Encode(): index=114 code=0010
inside Encode(): index=32 code=111
inside Encode(): index=111 code=1000
inside Encode(): index=117 code=011110
inside Encode(): index=116 code=1101
inside Encode(): index=32 code=111
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
inside Encode(): index=116 code=1101
inside Encode(): index=111 code=1000
inside Encode(): index=32 code=111
inside Encode(): index=116 code=1101
inside Encode(): index=104 code=0011
inside Encode(): index=101 code=010
inside Encode(): index=32 code=111
inside Encode(): index=71 code=100111000
inside Encode(): index=117 code=011110
inside Encode(): index=108 code=01110
inside Encode(): index=102 code=110010
inside Encode(): index=32 code=111
inside Encode(): index=83 code=000101101
inside Encode(): index=116 code=1101
inside Encode(): index=114 code=0010
inside Encode(): index=101 code=010
inside Encode(): index=97 code=1010
inside Encode(): index=109 code=1100111
inside Encode(): index=44 code=100100
inside Encode(): index=32 code=111
inside Encode(): index=13 code=100101
inside Encode(): index=10 code=100110
inside Encode(): index=110 code=0110
inside Encode(): index=111 code=1000
inside Encode(): index=114 code=0010
inside Encode(): index=116 code=1101
inside Encode(): index=104 code=0011
inside Encode(): index=32 code=111
inside Encode(): index=111 code=1000
inside Encode(): index=102 code=110010
inside Encode(): index=32 code=111
inside Encode(): index=67 code=000101100
inside Encode(): index=117 code=011110
inside Encode(): index=98 code=000100
inside Encode(): index=97 code=1010
inside Encode(): index=32 code=111
inside Encode(): index=105 code=0000
inside Encode(): index=110 code=0110
```