LAB 11

Name: Rahil acharya
RollNo. : CE004
ID: 20CEUOD004

AIM: Implement basic compression Techniques

1. Implement Arithmetic Coding and Decoding.

   a. Take the data set given in the pdf and find the codewords for GERMAN

   b. Decode the words from their respective codewords

main.m

```
clc;
clear all;
% Arithmetic coding
% calculating range_from and range_to
symbol = ['y' 'e' 'r' 'g' 'n' 'm' 'a' 'f' 'c'];
probability = [0.1 0.2 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
% symbol = ['a' 'b' 'c'];
% probability = [0.4 0.2 0.4];
n = length(symbol);
range_from = [];
range_to = [];
range_to_var = 0.0;
for i = 1 : n
    range_from(i) = range_to_var;
    range_to(i) = range_from(i)+probability(i);
    range_to_var = range_to(i);
end
tabD =
table(symbol(:),probability(:),range_from(:),range_to(:),'VariableNam
es',{'symbol','probability','range_from','range_to'});
disp(tabD)


% Now calculating LV HV & DIFF
% str = 'aacbc';
% INPUT
str = 'german';
strlen = length(str);
LV_old = 0; HV_old = 1; DIFF_old = 1;
LV = []; HV = []; DIFF = [];
```

```
for i = 1 : strlen
    idx = RF(tabD,str(i));
    LV(i) = LV_old + DIFF_old * tabD.range_from(idx);

    HV(i) = LV_old + DIFF_old * tabD.range_to(idx);

    DIFF(i) = HV(i)-LV(i);
    DIFF_old = DIFF(i);
    LV_old = LV(i);
    HV_old = HV(i);
end

tabDAC =
table(str(:),LV(:),HV(:),DIFF(:),'VariableNames',{'symbol','LV','HV',
'DIFF'});
disp(tabDAC)

% Decoding
input = 0.41735;
disp('Deconding in each line')

for i = 1 : 7
    idx = ranging(tabD,input);
    input = (input - tabD.range_from(idx)) / (tabD.range_to(idx) -
tabD.range_from(idx));
    disp(tabD.symbol(idx))
%     disp(input)
end
```

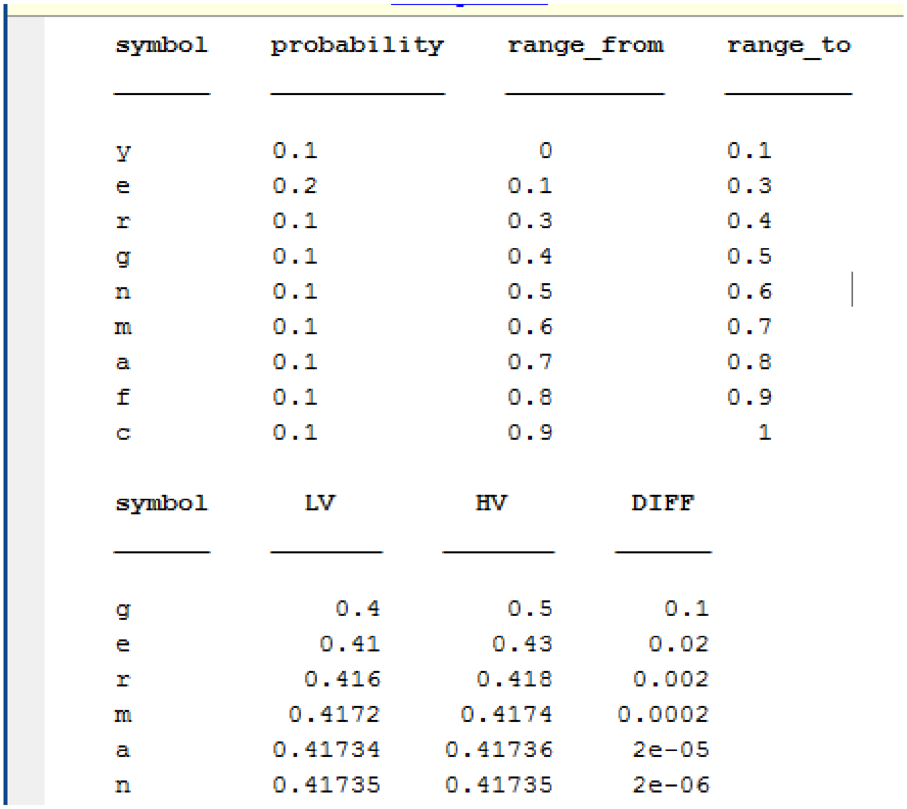
RF.m

```
function [i] = RF(table,symbol)
    [m,n] = size(table);
    for u = 1 : m
        charsym = table.symbol(u);
        if(charsym == symbol)
            i = u;
            break;
        end;
    end
end
```

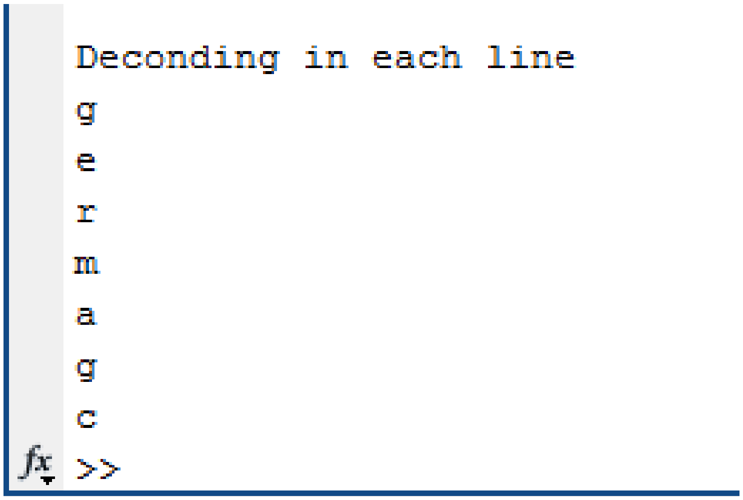
ranging.m

```
function [i] = ranging(table,val)
    [m,n] = size(table);
    for u = 1 : m
        v1 = table.range_from(u);
        v2 = table.range_to(u);
        if(val > v1 & val <= v2)
            i = u;
            break;
        end;
    end
end
```

Output:



```
symbol     probability     range_from     range_to
______     ___________     __________     ________

y          0.1               0            0.1
e          0.2             0.1            0.3
r          0.1             0.3            0.4
g          0.1             0.4            0.5
n          0.1             0.5            0.6
m          0.1             0.6            0.7
a          0.1             0.7            0.8
f          0.1             0.8            0.9
c          0.1             0.9              1

symbol       LV          HV          DIFF
______     _______     _______     ______

g              0.4         0.5        0.1
e             0.41        0.43       0.02
r            0.416       0.418      0.002
m           0.4172      0.4174     0.0002
a          0.41734     0.41736      2e-05
n          0.41735     0.41735      2e-06
```

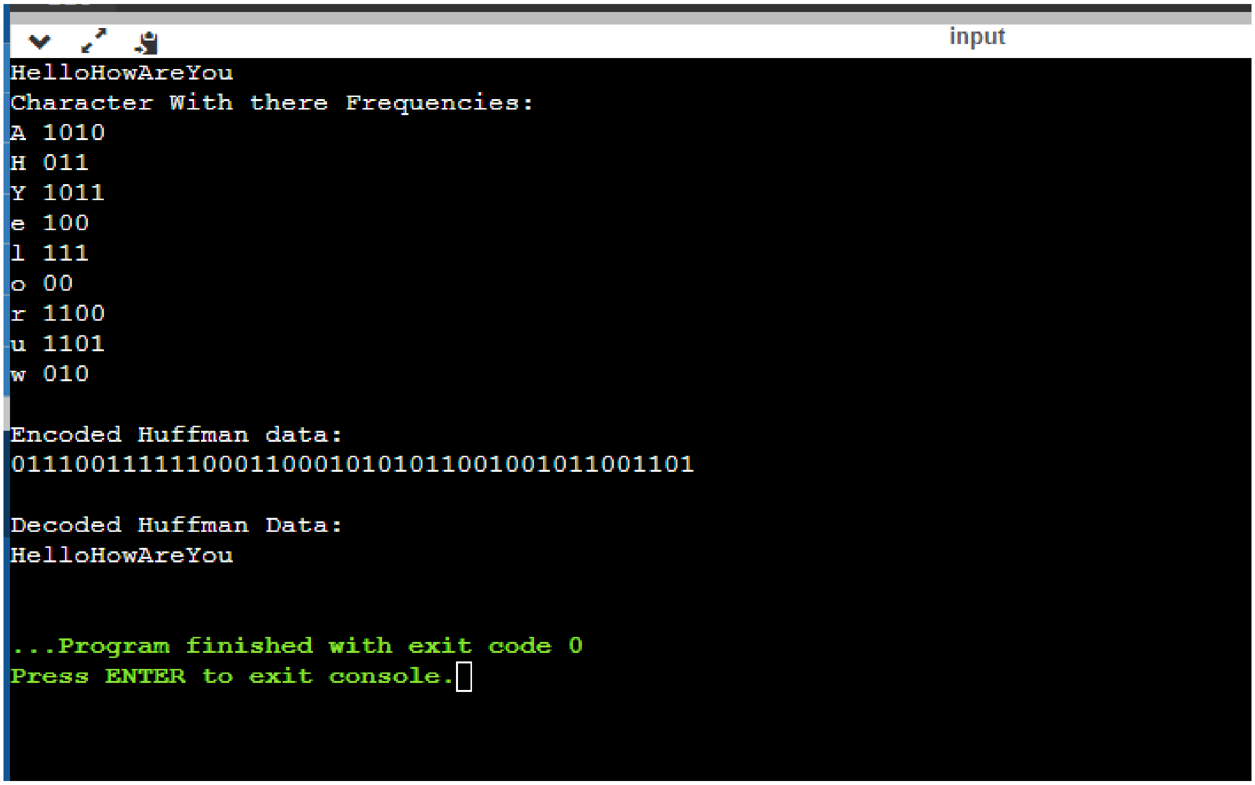


```
Deconding in each line
g
e
r
m
a
g
c
>>
```

2. Implement Huffman Coding

```
#include <bits/stdc++.h>
#define MAX_TREE_HT 256
using namespace std;

// to map each character its huffman value
map<char, string> codes;

// to store the frequency of character of the input data
map<char, int> freq;

// A Huffman tree node
struct MinHeapNode
{
    char data;             // One of the input characters
    int freq;             // Frequency of the character
    MinHeapNode *left, *right; // Left and right child

    MinHeapNode(char data, int freq)
    {
        left = right = NULL;
        this->data = data;
        this->freq = freq;
    }
};

// utility function for the priority queue
struct compare
{
    bool operator()(MinHeapNode* l, MinHeapNode* r)
    {
        return (l->freq > r->freq);
    }
};

// utility function to print characters along with
// there huffman value
void printCodes(struct MinHeapNode* root, string str)
{
    if (!root)
        return;
    if (root->data != '$')
        cout << root->data << ": " << str << "\n";
    printCodes(root->left, str + "0");
    printCodes(root->right, str + "1");
}

// utility function to store characters along with
// there huffman value in a hash table, here we
// have C++ STL map
void storeCodes(struct MinHeapNode* root, string str)
{
    if (root==NULL)
        return;
    if (root->data != '$')
        codes[root->data]=str;
    storeCodes(root->left, str + "0");
    storeCodes(root->right, str + "1");
}

// STL priority queue to store heap tree, with respect
// to their heap root node value
priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap;

// function to build the Huffman tree and store it
// in minHeap
void HuffmanCodes(int size)
{
    struct MinHeapNode *left, *right, *top;
    for (map<char, int>::iterator v=freq.begin(); v!=freq.end(); v++)
        minHeap.push(new MinHeapNode(v->first, v->second));
    while (minHeap.size() != 1)
    {
        left = minHeap.top();
        minHeap.pop();
        right = minHeap.top();
        minHeap.pop();
        top = new MinHeapNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        minHeap.push(top);
    }
    storeCodes(minHeap.top(), "");
}

// utility function to store map each character with its
// frequency in input string
void calcFreq(string str, int n)
{
    for (int i=0; i<str.size(); i++)
        freq[str[i]]++;
}

// function iterates through the encoded string s
// if s[i]=='1' then move to node->right
// if s[i]=='0' then move to node->left
// if leaf node append the node->data to our output string
string decode_file(struct MinHeapNode* root, string s)
{
    string ans = "";
    struct MinHeapNode* curr = root;
    for (int i=0;i<s.size();i++)
    {
        if (s[i] == '0')
        curr = curr->left;
        else
        curr = curr->right;

        // reached leaf node
        if (curr->left==NULL and curr->right==NULL)
        {
            ans += curr->data;
            curr = root;
        }
    }
    // cout<<ans<<endl;
    return ans+'\0';
}

// Driver program to test above functions
int main()
{
    string str;
    cin>>str;
    string encodedString, decodedString;
    calcFreq(str, str.length());
    HuffmanCodes(str.length());
    cout << "Character With there Frequencies:\n";
    for (auto v=codes.begin(); v!=codes.end(); v++)
        cout << v->first <<' ' << v->second << endl;
    for (auto i: str)
        encodedString+=codes[i];

    cout << "\nEncoded Huffman data:\n" << encodedString << endl;

    decodedString = decode_file(minHeap.top(), encodedString);
    cout << "\nDecoded Huffman Data:\n" << decodedString << endl;
    return 0;
}
```

Output:



```
HelloHowAreYou
Character With there Frequencies:
A 1010
H 011
Y 1011
e 100
l 111
o 00
r 1100
u 1101
w 010

Encoded Huffman data:
0111001111110001100010101011001001011001101

Decoded Huffman Data:
HelloHowAreYou


...Program finished with exit code 0
Press ENTER to exit console.
```