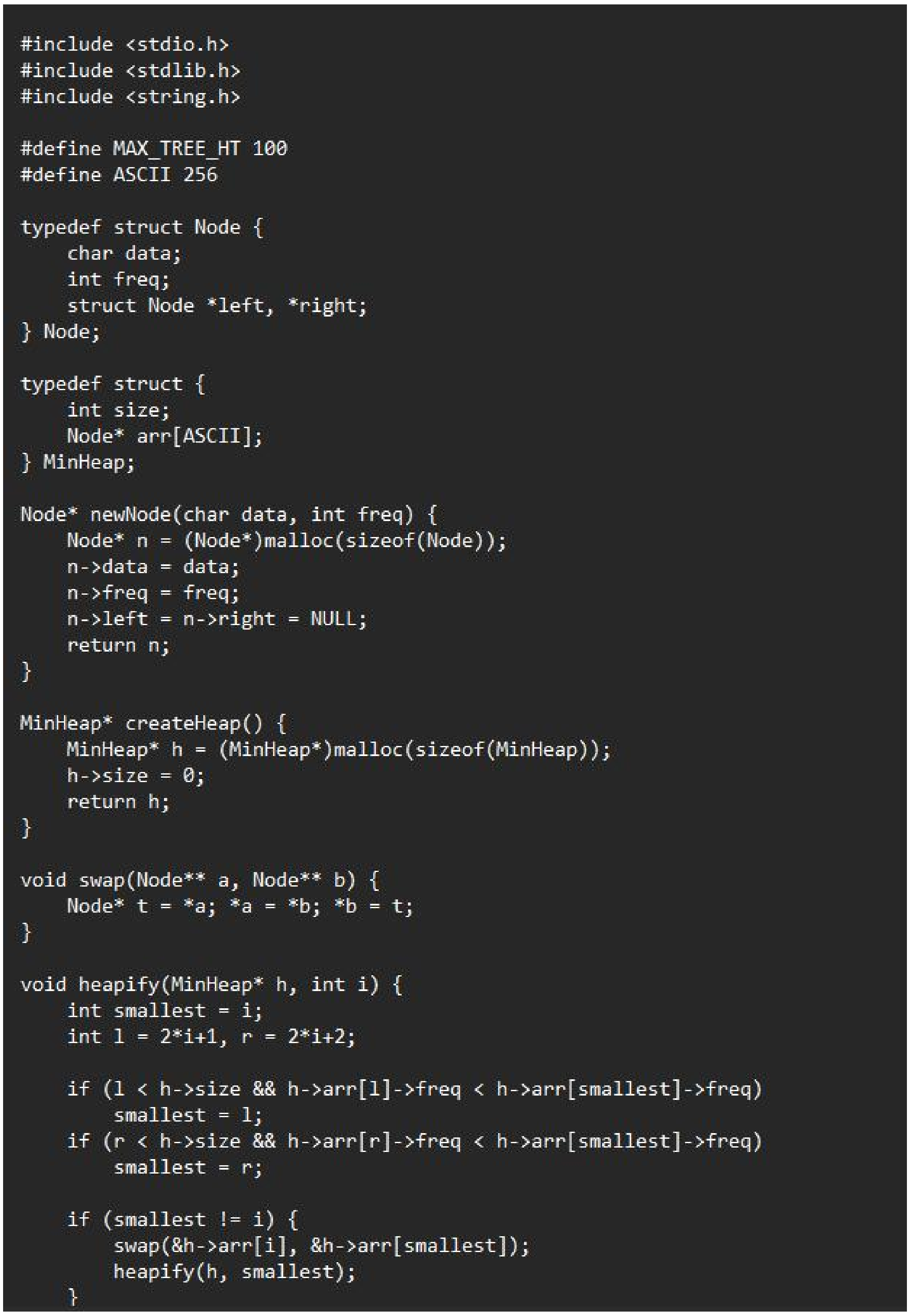
WEEK-6

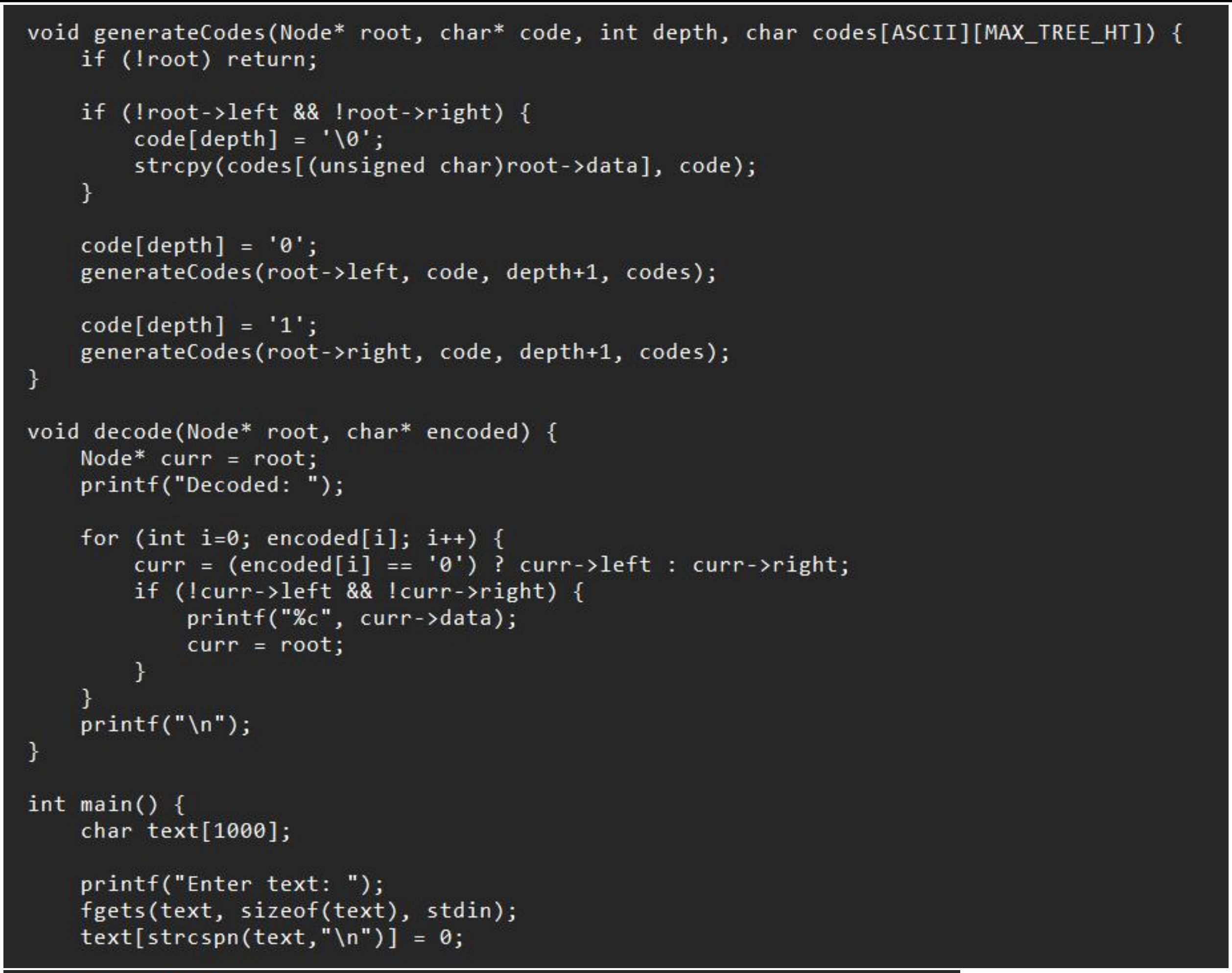
Name: B Lohith Krishnan
Roll Number: CH.SC.U4CSE24153

Huffman Coding:

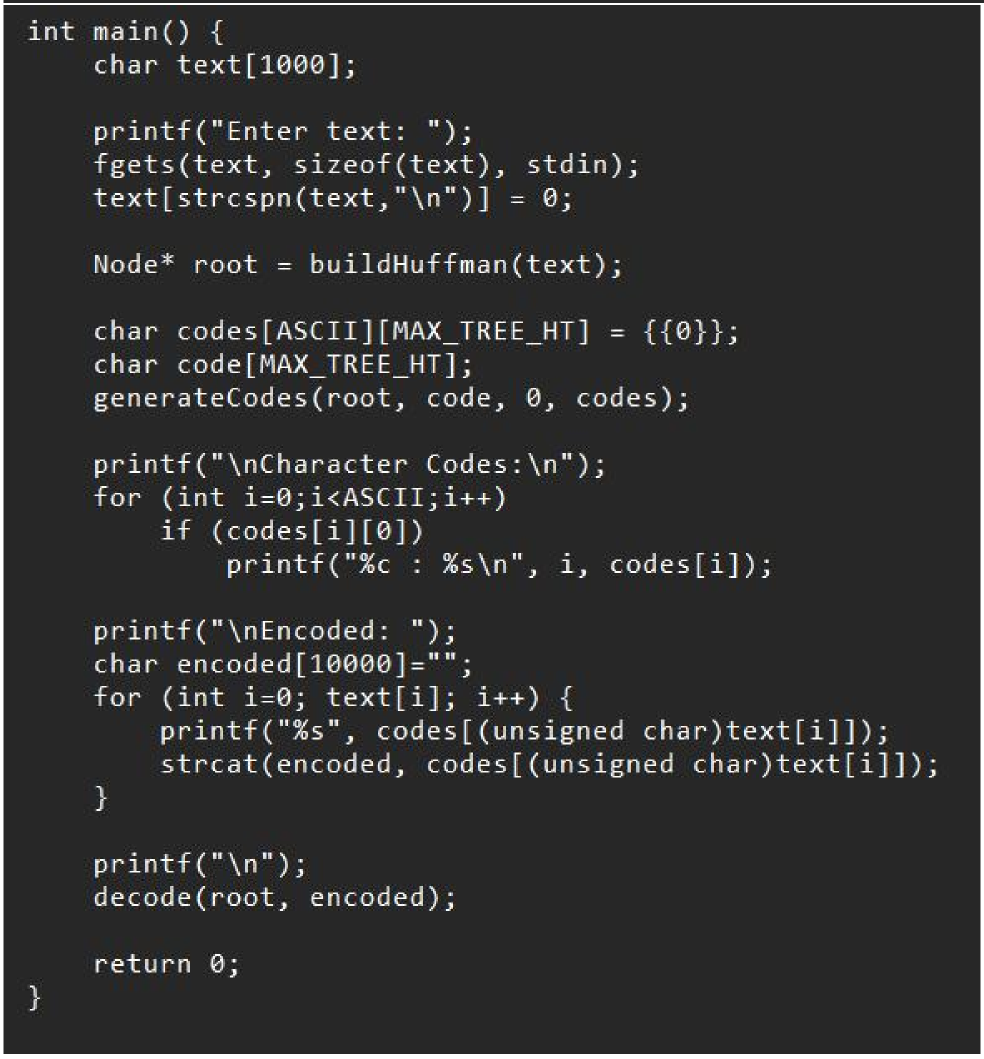
Code:



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_TREE_HT 100
#define ASCII 256

typedef struct Node {
    char data;
    int freq;
    struct Node *left, *right;
} Node;

typedef struct {
    int size;
    Node* arr[ASCII];
} MinHeap;

Node* newNode(char data, int freq) {
    Node* n = (Node*)malloc(sizeof(Node));
    n->data = data;
    n->freq = freq;
    n->left = n->right = NULL;
    return n;
}

MinHeap* createHeap() {
    MinHeap* h = (MinHeap*)malloc(sizeof(MinHeap));
    h->size = 0;
    return h;
}

void swap(Node** a, Node** b) {
    Node* t = *a; *a = *b; *b = t;
}

void heapify(MinHeap* h, int i) {
    int smallest = i;
    int l = 2*i+1, r = 2*i+2;

    if (l < h->size && h->arr[l]->freq < h->arr[smallest]->freq)
        smallest = l;
    if (r < h->size && h->arr[r]->freq < h->arr[smallest]->freq)
        smallest = r;

    if (smallest != i) {
        swap(&h->arr[i], &h->arr[smallest]);
        heapify(h, smallest);
    }
    if (smallest != i) {
        swap(&h->arr[i], &h->arr[smallest]);
        heapify(h, smallest);
    }
}

void insertHeap(MinHeap* h, Node* n) {
    int i = h->size++;
    while (i && n->freq < h->arr[(i-1)/2]->freq) {
        h->arr[i] = h->arr[(i-1)/2];
        i = (i-1)/2;
    }
    h->arr[i] = n;
}

Node* extractMin(MinHeap* h) {
    Node* temp = h->arr[0];
    h->arr[0] = h->arr[--h->size];
    heapify(h, 0);
    return temp;
}

Node* buildHuffman(char* text) {
    int freq[ASCII] = {0};
    for (int i=0; text[i]; i++)
        freq[(unsigned char)text[i]]++;

    MinHeap* h = createHeap();

    for (int i=0; i<ASCII; i++)
        if (freq[i])
            insertHeap(h, newNode(i, freq[i]));

    while (h->size > 1) {
        Node* left = extractMin(h);
        Node* right = extractMin(h);
        Node* top = newNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        insertHeap(h, top);
    }
    return extractMin(h);
}

void generateCodes(Node* root, char* code, int depth, char codes[ASCII][MAX_TREE_HT]) {
    if (!root) return;
```

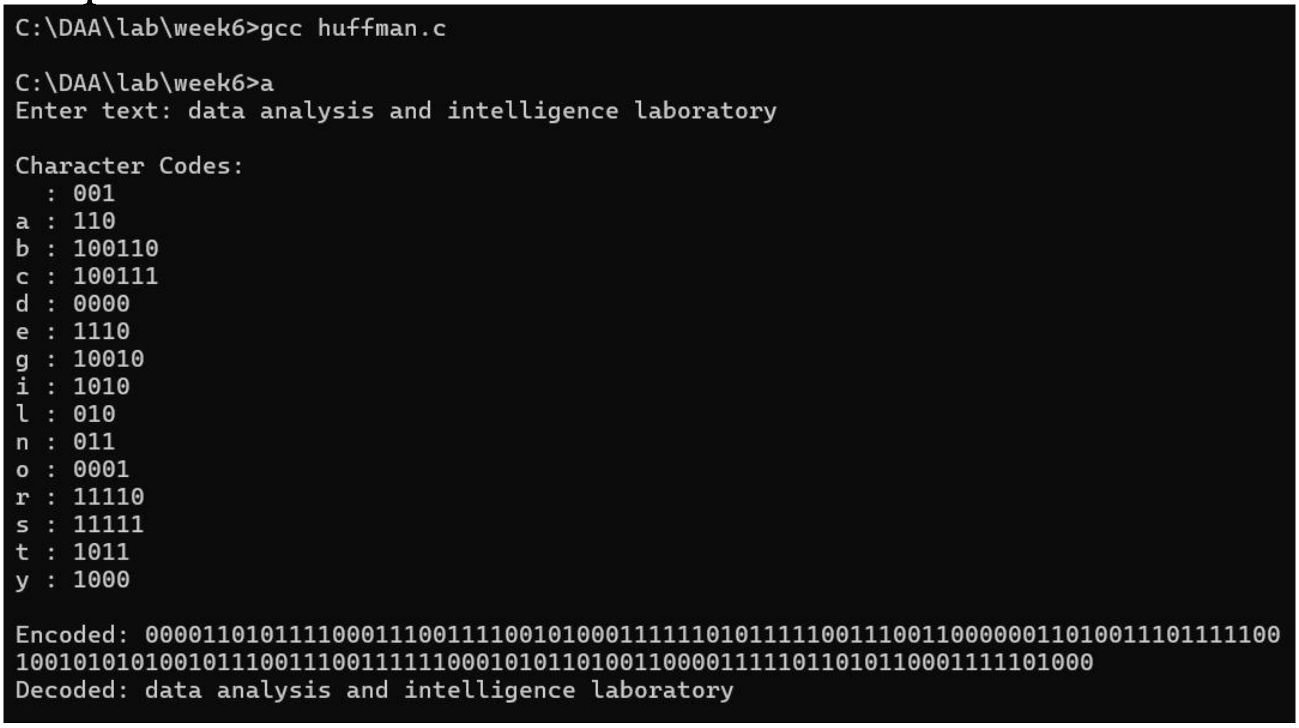


```c
void generateCodes(Node* root, char* code, int depth, char codes[ASCII][MAX_TREE_HT]) {
    if (!root) return;

    if (!root->left && !root->right) {
        code[depth] = '\0';
        strcpy(codes[(unsigned char)root->data], code);
    }

    code[depth] = '0';
    generateCodes(root->left, code, depth+1, codes);

    code[depth] = '1';
    generateCodes(root->right, code, depth+1, codes);
}

void decode(Node* root, char* encoded) {
    Node* curr = root;
    printf("Decoded: ");

    for (int i=0; encoded[i]; i++) {
        curr = (encoded[i] == '0') ? curr->left : curr->right;
        if (!curr->left && !curr->right) {
            printf("%c", curr->data);
            curr = root;
        }
    }
    printf("\n");
}

int main() {
    char text[1000];

    printf("Enter text: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text,"\n")] = 0;
```



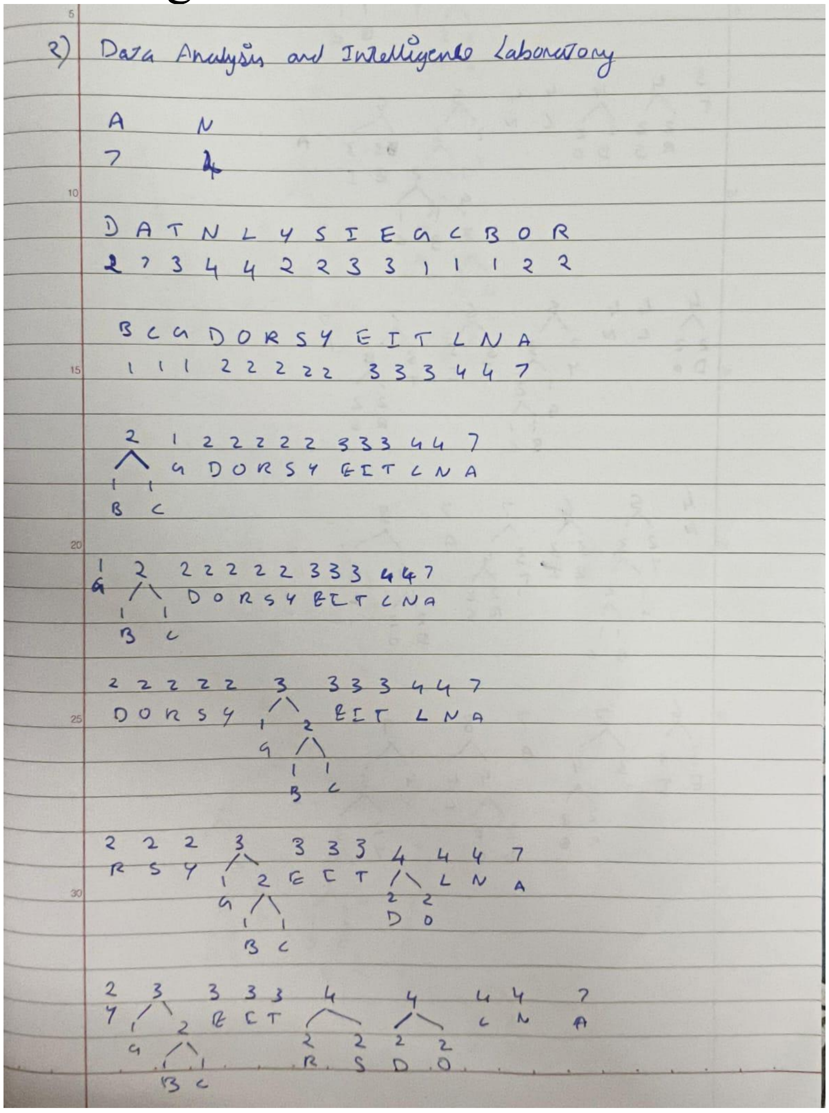
```c
int main() {
    char text[1000];

    printf("Enter text: ");
    fgets(text, sizeof(text), stdin);
    text[strcspn(text,"\n")] = 0;

    Node* root = buildHuffman(text);

    char codes[ASCII][MAX_TREE_HT] = {{0}};
    char code[MAX_TREE_HT];
    generateCodes(root, code, 0, codes);

    printf("\nCharacter Codes:\n");
    for (int i=0;i<ASCII;i++)
        if (codes[i][0])
            printf("%c : %s\n", i, codes[i]);

    printf("\nEncoded: ");
    char encoded[10000]="";
    for (int i=0; text[i]; i++) {
        printf("%s", codes[(unsigned char)text[i]]);
        strcat(encoded, codes[(unsigned char)text[i]]);
    }

    printf("\n");
    decode(root, encoded);

    return 0;
}
```

Output:

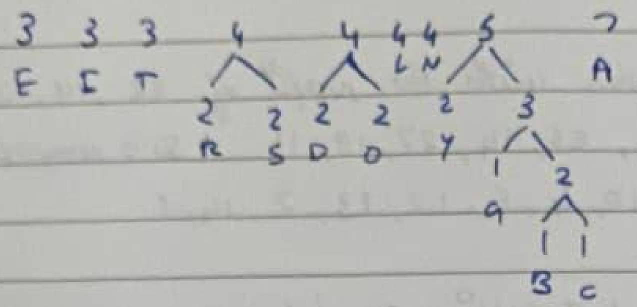


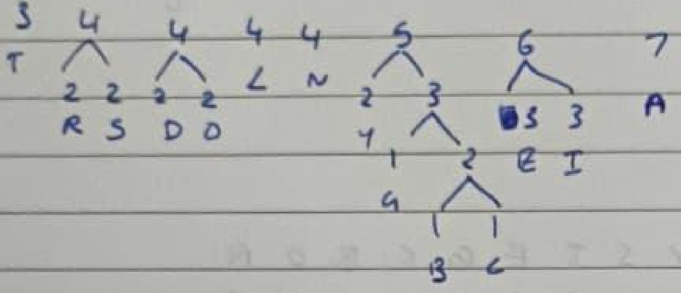
```
C:\DAA\lab\week6>gcc huffman.c

C:\DAA\lab\week6>a
Enter text: data analysis and intelligence laboratory

Character Codes:
  : 001
a : 110
b : 100110
c : 100111
d : 0000
e : 1110
g : 10010
i : 1010
l : 010
n : 011
o : 0001
r : 11110
s : 11111
t : 1011
y : 1000

Encoded: 00001101011110001110011110010100011111101011111001110011000000110100111011111100
1001010101001011100111001111110001010110100110000111110110101100011111101000
Decoded: data analysis and intelligence laboratory
```

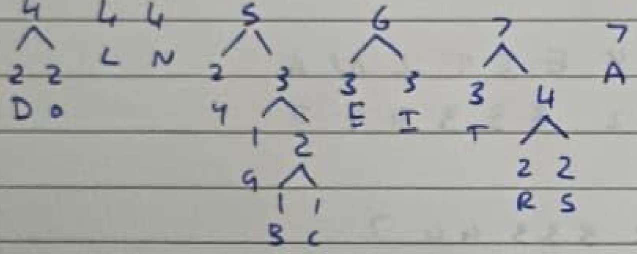
Working:

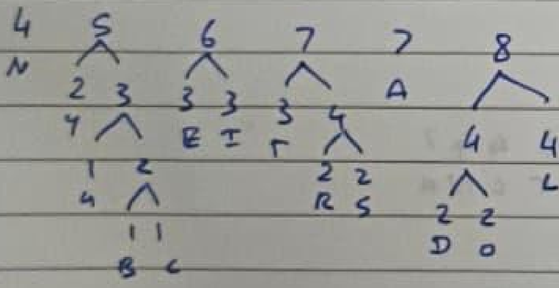

2) Data Analysis and Intelligence Laboratory
A 7
N 4
D A T N L Y S I E G C B O R
2 7 3 4 4 2 2 3 3 1 1 1 2 2
B C G D O R S Y E I T L N A
1 1 1 2 2 2 2 2 3 3 3 4 4 7

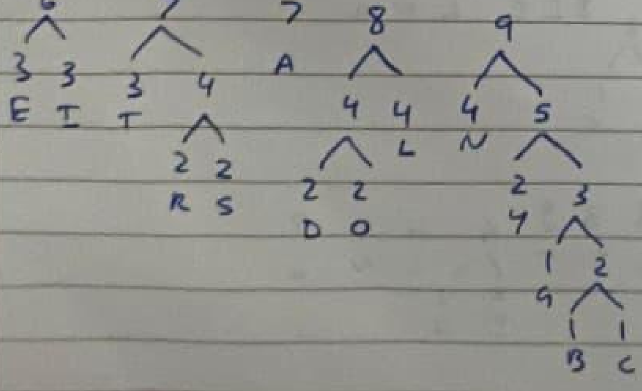
Date :


3 3 3 4 4 4 4 5 7
E I T L N A
2 2 2 2 2 3
R S D O Y
1 2
G
1 1
B C


3 4 4 4 4 5 6 7
T L N A
2 2 2 2 2 3 3 3
R S D O Y E I
1 2
G
1 1
B C


4 4 4 5 6 7 7
L N A
2 2 2 3 3 3 3 4
D O Y E I T
1 2 2 2
G R S
1 1
B C


4 5 6 7 7 8
N A
2 3 3 3 3 4 4 4
Y E I T L
1 2 2 2 2 2
G R S D O
1 1
B C


6 7 7 8 9
A
3 3 3 4 4 4 4 5
E I T L N
2 2 2 2 2 3
R S D O Y
1 2
G
1 1
B C