

# Lab 9

Prasanna Natarajan  
1410110298

Code:

```
/*
Name   : Prasanna Natarajan
Roll Number   : 1410110298
Inputs  : n = number of characters
Outputs : The huffman code for the given sequence
*/

#include<stdio.h>
#include<stdlib.h>
#include<time.h>

#define n 100

typedef struct node{
    int freq;
    char c;
    struct node *left;
    struct node *right;
}node_t;

char input[n];
int counter[4];
node_t ** array;

// function defenitions
void generate();
void frequency_counter();
node_t* generate_code(node_t *root);
node_t* create_node(int elem, char ch);
node_t* insert_node(node_t *root, node_t *node);
node_t** sort_node(int size);

int main(){

    generate();
    frequency_counter();
    int i;
    array = malloc(sizeof(struct node)*4);
```

```

for(i=0;i<4;i++){

    array[i] = create_node(counter[i], 'a'+i);
    //printf("%d",array[i]->freq);fflush(stdout);
}
puts("here");
array = sort_node(4);
puts("here");
node_t *root = NULL;
puts("here");
//node_t **arr = array;
root = generate_code(root);
return 0;
}

```

```

node_t** sort_node(int size){
    node_t *temp;
    int i,j;
    for(i=0;i<size-1;i++){
        for(j=0;j<size-i-1;j++){
            if(array[j]->freq >= array[j+1]->freq){
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }
}

```

```

void generate(){
    int i;
    time_t t;
    srand((unsigned) time(&t));

    for(i=0;i<n;i++){
        input[i] = 'a'+(rand()%4);
        //printf("%c ",input[i]);
    }
}

```

```

void frequency_counter(){
    int i;

```

```

        for(i=0;i<n;i++){
            counter[input[i]-'a']++;
        }
        /*
        for(i=0;i<4;i++)
            printf("%d ",counter[i]);
        */
    }
node_t *create_node(int elem, char ch)
{
    node_t *node;

    node = (node_t *) malloc(sizeof(node_t));
    if (node == NULL)
    {
        fprintf(stderr, "malloc failed\n");
        exit(1);
    }
    node->freq = elem;
    node->left = NULL;
    node->right = NULL;
    node->c = ch;

    return node;
}
node_t* generate_code(node_t *root){
    int i,j;
    for(i=0;i<4-1;i++){
        node_t *z = create_node(0,'a');
        array = sort_node(4-i);
        z->left = array[i];
        z->right = array[i+1];
        z->freq = array[i]->freq + array[i+1]->freq;
        root = insert_node(root,z);
        array[0] = z;
    }

    return root;
}

/* Recursive variant of insert function */
node_t* insert_node(node_t *root, node_t *node)

```

```

{
    /* Handle empty tree as a special case */
    if (root == NULL)
    {
        root = node;
        return root;
    }

    if (node->freq < root->freq)
        root->left = insert_node(root->left, node);
    else
        root->right = insert_node(root->right, node);
    return root;
}

```

The above code is not working. It gives the following error

```

prasananna@LENOVO-PC: /mnt/c/Users/prasananna/Documents/Studies/Semester 6/Algorithms/labs/lab9
Starting program: /mnt/c/Users/prasananna/Documents/Studies/Semester 6/Algorithms/labs/lab9/a.out
warning: Error disabling address space randomization: Success
warning: linux_ptrace_test_ret_to_nx: PTRACE_KILL waitpid returned -1: Interrupted system call
here
here
here

Program received signal SIGSEGV, Segmentation fault.
0x000000000040081a in sort_node (size=100) at lab9.c:59
59      if(array[j]->freq >= array[j+1]->freq){
(gdb) print j
$1 = 0
(gdb) print array[j]
Cannot access memory at address 0x3
(gdb) print array
$2 = (node_t **) 0x3
(gdb)

```