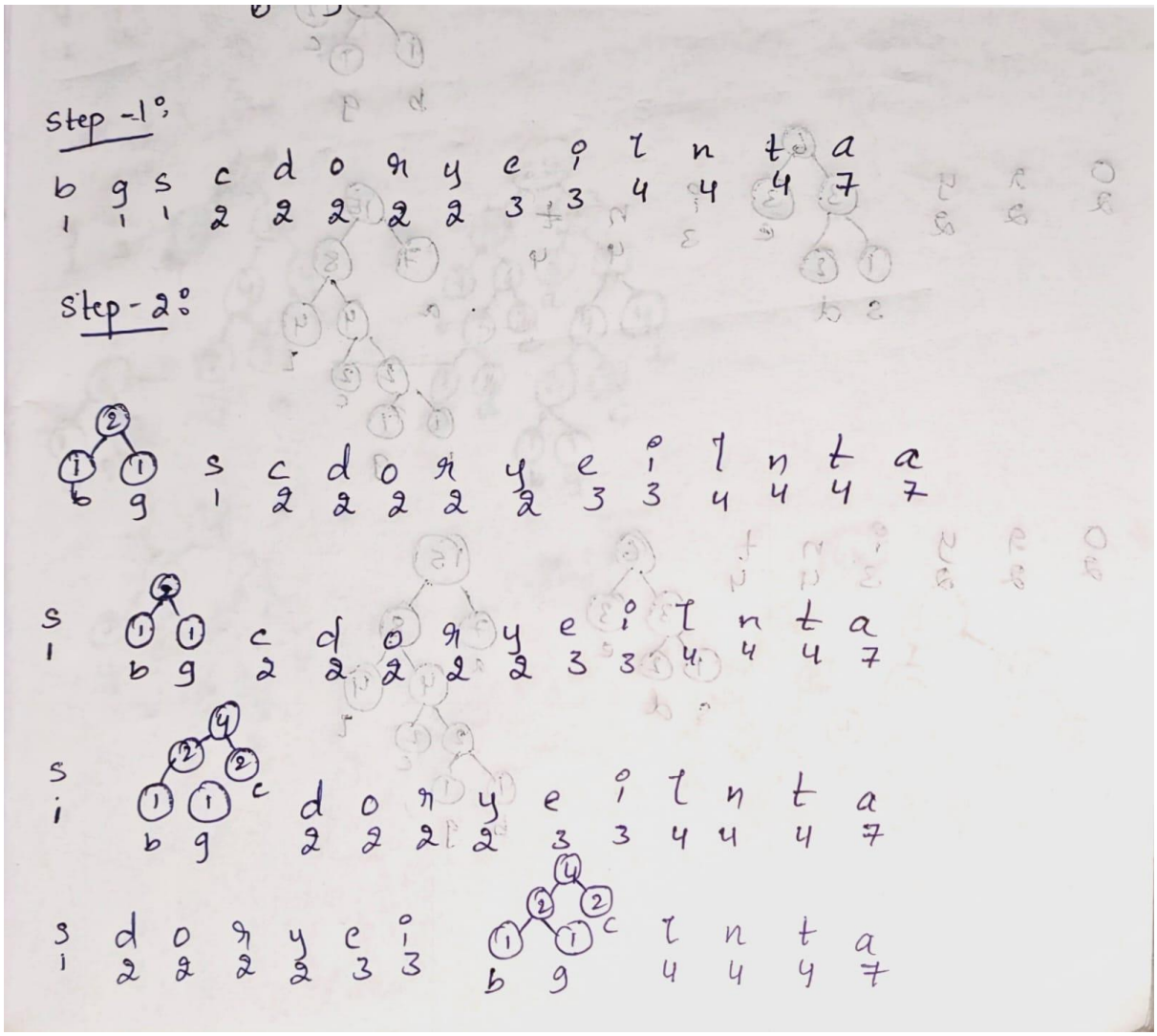
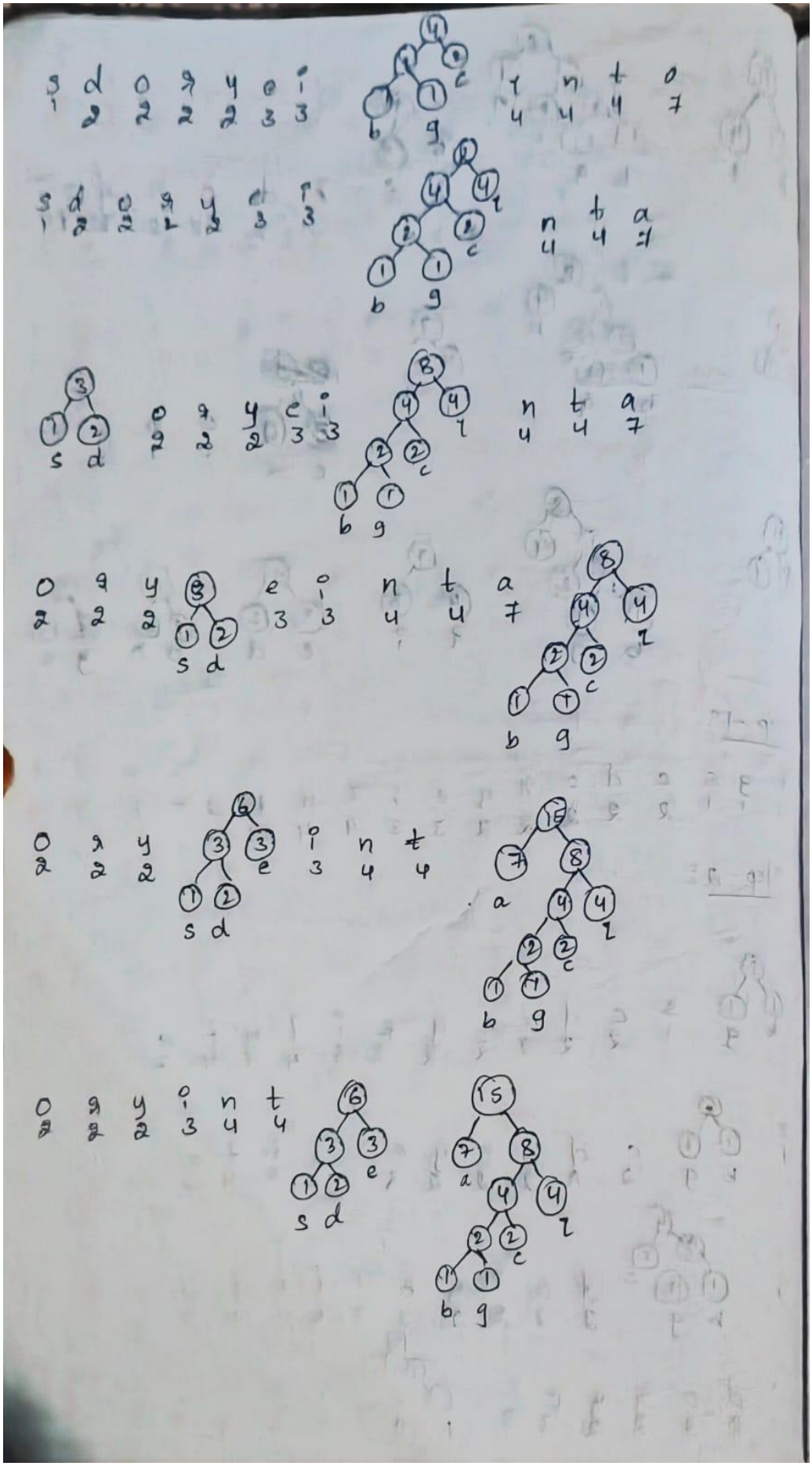
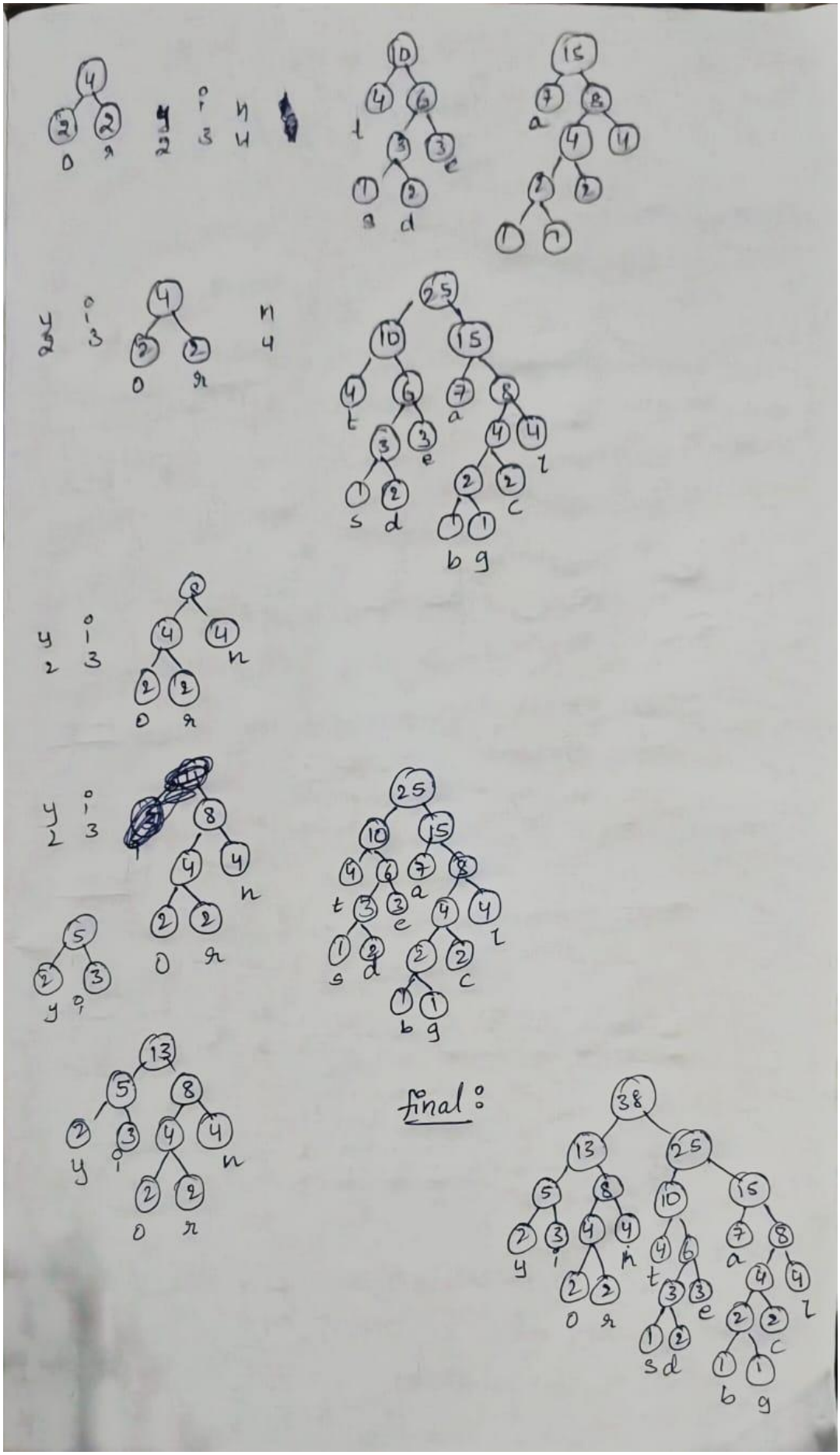
WEEK – 7

LOGIC :


step -1:
b g s c d o r y e i l n t a
1 1 1 2 2 2 2 2 3 3 4 4 4 7
step - 2:
s c d o r y e i l n t a
1 2 2 2 2 2 3 3 4 4 4 7
s c d o r y e i l n t a
1 2 2 2 2 2 3 3 4 4 4 7
s d o r y e i l n t a
i 2 2 2 2 3 3 4 4 4 7




final:

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_TREE_HT 100
struct MinHeapNode {
    char data;
    unsigned freq;
    struct MinHeapNode *left, *right;
};
struct MinHeap {
    unsigned size;
    unsigned capacity;
    struct MinHeapNode** array;
};
struct MinHeapNode* newNode(char data, unsigned freq) {
    struct MinHeapNode* temp =
        (struct MinHeapNode*)malloc(sizeof(struct MinHeapNode));
    temp->left = temp->right = NULL;
    temp->data = data;
    temp->freq = freq;
    return temp;
}
struct MinHeap* createMinHeap(unsigned capacity) {
    struct MinHeap* minHeap =
        (struct MinHeap*)malloc(sizeof(struct MinHeap));
    minHeap->size = 0;
    minHeap->capacity = capacity;
    minHeap->array =
        (struct MinHeapNode**)malloc(capacity * sizeof(struct MinHeapNode*));
    return minHeap;
}
void swapMinHeapNode(struct MinHeapNode** a,
                     struct MinHeapNode** b) {
    struct MinHeapNode* t = *a;
    *a = *b;
    *b = t;
}
void minHeapify(struct MinHeap* minHeap, int idx) {
    int smallest = idx;
    int left = 2 * idx + 1;
    int right = 2 * idx + 2;

    if (left < minHeap->size &&
        minHeap->array[left]->freq <
        minHeap->array[smallest]->freq)
```

```
45          smallest = left;
46
47      if (right < minHeap->size &&
48          minHeap->array[right]->freq <
49          minHeap->array[smallest]->freq)
50          smallest = right;
51
52      if (smallest != idx) {
53          swapMinHeapNode(&minHeap->array[smallest],
54                          &minHeap->array[idx]);
55          minHeapify(minHeap, smallest);
56      }
57  }
58  struct MinHeapNode* extractMin(struct MinHeap* minHeap) {
59      struct MinHeapNode* temp = minHeap->array[0];
60      minHeap->array[0] =
61          minHeap->array[minHeap->size - 1];
62      --minHeap->size;
63      minHeapify(minHeap, 0);
64      return temp;
65  }
66  void insertMinHeap(struct MinHeap* minHeap,
67                     struct MinHeapNode* minHeapNode) {
68      ++minHeap->size;
69      int i = minHeap->size - 1;
70
71      while (i && minHeapNode->freq <
72             minHeap->array[(i - 1) / 2]->freq) {
73          minHeap->array[i] =
74              minHeap->array[(i - 1) / 2];
75          i = (i - 1) / 2;
76      }
77
78      minHeap->array[i] = minHeapNode;
79  }
80  struct MinHeapNode* buildHuffmanTree(char data[],
81                                       int freq[],
82                                       int size) {
83      struct MinHeapNode *left, *right, *top;
84      struct MinHeap* minHeap = createMinHeap(size);
85
86      for (int i = 0; i < size; ++i)
```

```
        minHeap->array[i] = newNode(data[i], freq[i]);

    minHeap->size = size;

    while (minHeap->size != 1) {
        left = extractMin(minHeap);
        right = extractMin(minHeap);

        top = newNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;

        insertMinHeap(minHeap, top);
    }

    return extractMin(minHeap);
}
void printCodes(struct MinHeapNode* root,
                int arr[], int top) {
    if (root->left) {
        arr[top] = 0;
        printCodes(root->left, arr, top + 1);
    }

    if (root->right) {
        arr[top] = 1;
        printCodes(root->right, arr, top + 1);
    }

    if (!root->left && !root->right) {
        printf("%c: ", root->data);
        for (int i = 0; i < top; ++i)
            printf("%d", arr[i]);
        printf("\n");
    }
}
int main() {
    char arr[] = {'a', 'b', 'c', 'd', 'e', 'f'};
    int freq[] = {5, 9, 12, 13, 16, 45};

    int size = sizeof(arr) / sizeof(arr[0]);

```

```
129     struct MinHeapNode* root =
130         buildHuffmanTree(arr, freq, size);
131
132     int code[MAX_TREE_HT], top = 0;
133
134     printf("Huffman Codes:\n");
135     printCodes(root, code, top);
136
137     return 0;
138 }
```

OUTPUT:

```
Huffman Codes:
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111

--------------------------------
Process exited after 0.08794 seconds with return value 0
Press any key to continue . . .
```