Nikit Gokhe
Comp D1
Roll no. 324022
GR no. 21810522

ASSIGNMENT

HUFFMAN CODING

SOURCE CODE:

```
#include <iostream>
#include <cstdlib>
using namespace std;

// calculating height of Huffman Tree
#define MAX_TREE_HT 100

struct MinHeapNode {

    char data;
    int freq;
    struct MinHeapNode *left, *right;
};

struct MinHeap {

    unsigned size;
    unsigned capacity;
    struct MinHeapNode** array;   // Attay of minheap node pointers
};


struct MinHeapNode* newNode(char data, int 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(minHeap->
capacity * sizeof(struct MinHeapNode*));
    return minHeap;
}


// swap two min heap nodes
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)
        smallest = left;
```

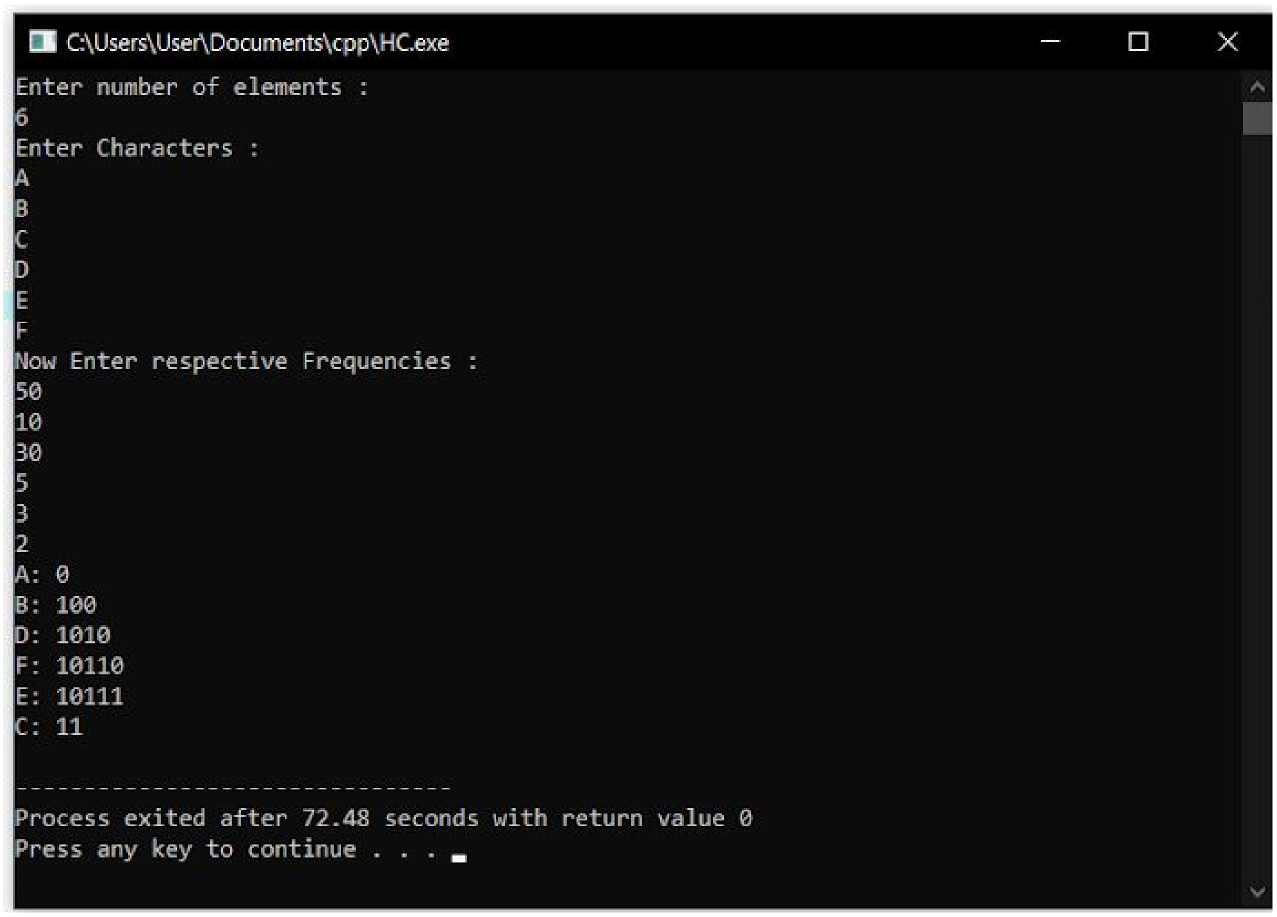
```
   if (right < minHeap->size && minHeap->array[right]->
freq < minHeap->array[smallest]->freq)
      smallest = right;

   if (smallest != idx) {
      swapMinHeapNode(&minHeap->array[smallest],
                  &minHeap->array[idx]);
      minHeapify(minHeap, smallest);
   }
}

int isSizeOne(struct MinHeap* minHeap)
{
        return (minHeap->size == 1);
}

struct MinHeapNode* extractMin(struct MinHeap* minHeap)
{
        struct MinHeapNode* temp = minHeap->array[0];
   minHeap->array[0]
      = minHeap->array[minHeap->size - 1];

   --minHeap->size;
   minHeapify(minHeap, 0);

   return temp;
}


void insertMinHeap(struct MinHeap* minHeap, struct MinHeapNode* minHeapNode)
{
        ++minHeap->size;
   int i = minHeap->size - 1;

   while (i && minHeapNode->freq < minHeap->array[(i - 1) / 2]->freq) {

      minHeap->array[i] = minHeap->array[(i - 1) / 2];
      i = (i - 1) / 2;
   }

   minHeap->array[i] = minHeapNode;
}
```

```
  // A standard function to build min heap
void buildMinHeap(struct MinHeap* minHeap)
{
        int n = minHeap->size - 1;
   int i;

   for (i = (n - 1) / 2; i >= 0; --i)
      minHeapify(minHeap, i);
}

void printArr(int arr[], int n)
{
   int i;
   for (i = 0; i < n; ++i)
      cout<< arr[i];

   cout<<"\n";
}

int isLeaf(struct MinHeapNode* root)
{

   return !(root->left) && !(root->right);
}

struct MinHeap* createAndBuildMinHeap(char data[], int freq[], int size)
{

   struct MinHeap* minHeap = createMinHeap(size);

   for (int i = 0; i < size; ++i)
      minHeap->array[i] = newNode(data[i], freq[i]);

   minHeap->size = size;
   buildMinHeap(minHeap);

   return minHeap;
}
```

```
// The main function that builds Huffman tree
struct MinHeapNode* buildHuffmanTree(char data[], int freq[], int size)

{
    struct MinHeapNode *left, *right, *top;
    struct MinHeap* minHeap = createAndBuildMinHeap(data, freq, size);

    while (!isSizeOne(minHeap)) {

        left = extractMin(minHeap);
        right = extractMin(minHeap);

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

        top->left = left;
        top->right = right;

        insertMinHeap(minHeap, top);
    }

    return extractMin(minHeap);
}

// Prints huffman codes from the root of Huffman Tree.
// It uses arr[] to store codes
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 (isLeaf(root)) {
        cout<< root->data <<": ";
        printArr(arr, top);
    }
}


void HuffmanCodes(char data[], int freq[], int size)
{
    struct MinHeapNode* root
        = buildHuffmanTree(data, freq, size);   // Construct Huffman Tree
    int arr[MAX_TREE_HT], top = 0;

    printCodes(root, arr, top);
}

// Driver program to test above functions
int main()
{
        int n;
        cout<<"Enter number of elements :\n";
        cin>>n;

    char arr[n];
    int freq[n];
    cout<<"Enter Characters :\n";
    for(int i=0; i<n; i++)
    {
        cin>>arr[i];
        }

        cout<<"Now Enter respective Frequencies :\n";
        for(int i=0; i<n; i++)
    {
        cin>>freq[i];
        }

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

    return 0;
}
```

OUTPUT :

1.



C:\Users\User\Documents\cpp\HC.exe

```
Enter number of elements :
6
Enter Characters :
A
B
C
D
E
F
Now Enter respective Frequencies :
50
10
30
5
3
2
A: 0
B: 100
D: 1010
F: 10110
E: 10111
C: 11

--------------------------------
Process exited after 72.48 seconds with return value 0
Press any key to continue . . .
```