Name: Sushant S. Gawade

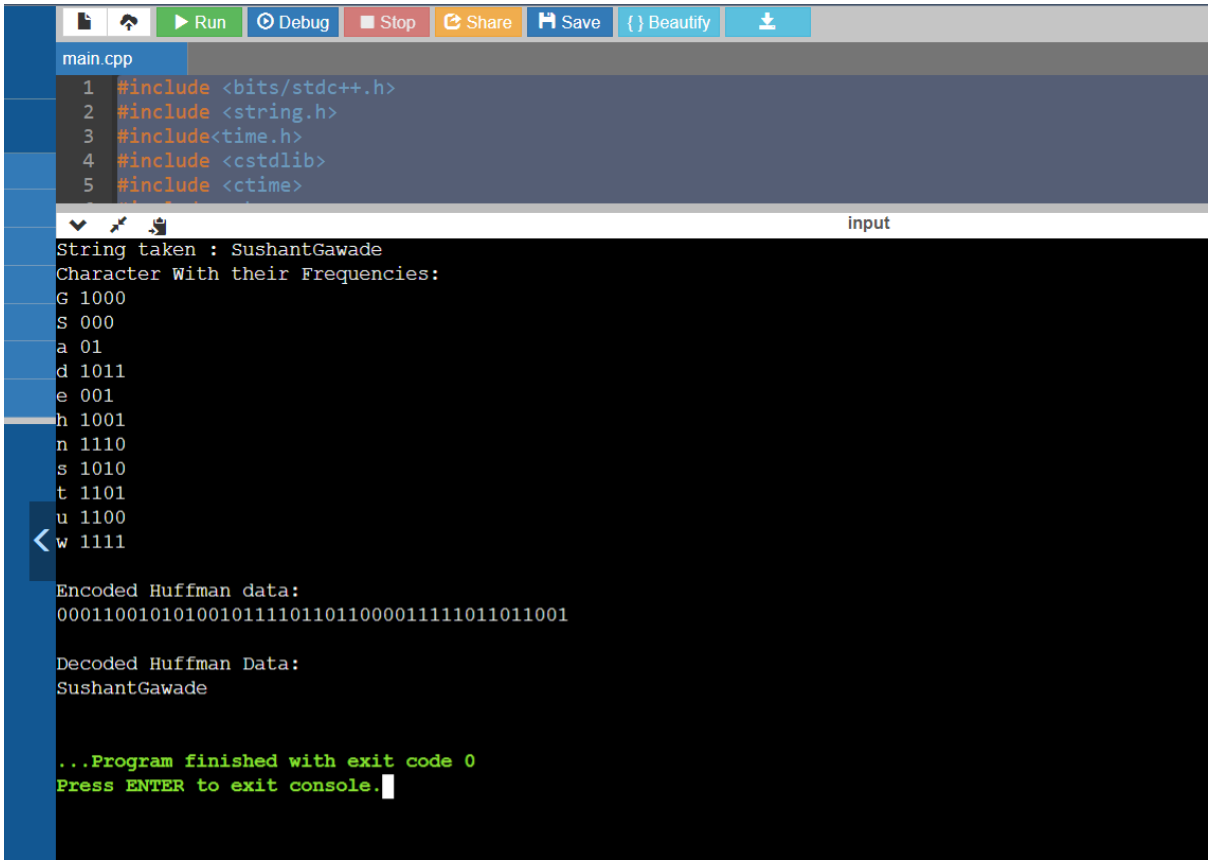
Roll No.: 118

PRN No.: 0120190104

Batch: B3

Practical 6: Design & Implement Huffman Algorithm using Greedy Approach. Calculate the time complexity of the algorithm.

Output:



Code:

```
#include <bits/stdc++.h>

#include <string.h>

#include<time.h>

#include <cstdlib>

#include <ctime>

#include <chrono>

using namespace std::chrono;

#define MAX_TREE_HT 256

using namespace std;

map<char, string> codes;

map<char, int> freq;

struct MinHeapNode

{

   char data;

   int freq;

   MinHeapNode *left, *right;


   MinHeapNode(char data, int freq)

   {

      left = right = NULL;

      this->data = data;

      this->freq = freq;

   }

};
```

```
struct compare
{
    bool operator()(MinHeapNode* l, MinHeapNode* r)
    {
        return (l->freq > r->freq);
    }
};
void printCodes(struct MinHeapNode* root, string str)
{
    if (!root)
        return;
    if (root->data != '$')
        cout << root->data << ": " << str << "\n";
    printCodes(root->left, str + "0");
    printCodes(root->right, str + "1");
}
void storeCodes(struct MinHeapNode* root, string str)
{
    if (root==NULL)
        return;
    if (root->data != '$')
        codes[root->data]=str;
    storeCodes(root->left, str + "0");
    storeCodes(root->right, str + "1");
}
```

```
priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> 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(), "");
}
void calcFreq(string str, int n)
{
   for (int i=0; i<str.size(); i++)
      freq[str[i]]++;
}
```

```
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;

        if (curr->left==NULL and curr->right==NULL)
        {
            ans += curr->data;
            curr = root;
        }
    }
    return ans+'\0';
}

int main()
{

    string str = "SushantGawade";
    cout<<"String taken : "<<str<<endl;
```

```
    string encodedString, decodedString;
    calcFreq(str, str.length());
    HuffmanCodes(str.length());

    cout << "Character With their 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;

}
```