view source

print?

```
001 //RIT2009061
002 //vikesh iiita
003 #include <iostream>
004 #include <fstream>
005 #include <vector>
006 #include <string>
007 #include <queue>
008 #include <algorithm>
009
010 using namespace std;
011
012 struct node {
013     int weight;
014     unsigned char value;
015     const node *child0;
016     const node *child1;
017
018     node( unsigned char c = 0, int i = -1 ) {
019         value = c;
020         weight = i;
021         child0 = 0;
022         child1 = 0;
023     }
024
025     node( const node* c0, const node *c1 ) {
026         value = 0;
027         weight = c0->weight + c1->weight;
028         child0 = c0;
029         child1 = c1;
030     }
031
032     bool operator<( const node &a ) const {
033         return weight >a.weight;
034     }
035
036     void traverse(string code="") const {
037
038     if ( child0 ) {
```

```
039         child0->traverse( code + '0' );
040         child1->traverse( code + '1' );
041     } else {
042         cout <<" " <<value <<"      ";
043         cout <<weight;
044         cout <<"   " <<code <<endl;
045     }
046 }
047
048 };
049
050
051
052 void count_chars( int *counts )
053 {
054     for ( int i = 0 ; i <256 ; i++ )
055         counts[ i ] = 0;
056     ifstream file( "input.dat" );
057     if ( !file ) {
058         cout <<"Couldn't open the input file!\n";
059         throw "abort";
060     }
061     file.setf( ios::skipws );
062     for ( ; ; ) {
063         unsigned char c;
064         file>> c;
065         if ( file )
066             counts[ c ]++;
067         else
068             break;
069    }
070 }
071
072 int main()
073 {
074     int counts[ 256 ];
075     count_chars( counts );
076     priority_queue < node > q;
077
078     for ( int i = 0 ; i <256 ; i++ )
```

```
079         if ( counts[ i ] )
080             q.push( node( i, counts[ i ] ) );
081 
082     while ( q.size() >1 ) {
083         node *child0 = new node( q.top() );
084         q.pop();
085         node *child1 = new node( q.top() );
086         q.pop();
087         q.push( node( child0, child1 ) );
088     }
089 
090     cout <<"CHAR  FREQUENCY  HOFFMAN-CODE" <<endl;
091     q.top().traverse();
092     return 0;
093 }
094 
095 //RIT2009061
096 //vikesh iiita
097 #include <iostream>
098 #include <fstream>
099 #include <vector>
100 #include <string>
101 #include <queue>
102 #include <algorithm>
103 
104 using namespace std;
105 
106 struct node {
107     int weight;
108     unsigned char value;
109     const node *child0;
110     const node *child1;
111 
112     node( unsigned char c = 0, int i = -1 ) {
113         value = c;
114         weight = i;
115         child0 = 0;
116         child1 = 0;
117     }
118 
```

```
119     node( const node* c0, const node *c1 ) {
120         value = 0;
121         weight = c0->weight + c1->weight;
122         child0 = c0;
123         child1 = c1;
124     }
125
126     bool operator<( const node &a ) const {
127         return weight >a.weight;
128     }
129
130     void traverse(string code="") const {
131
132     if ( child0 ) {
133         child0->traverse( code + '0' );
134         child1->traverse( code + '1' );
135     } else {
136         cout <<" " <<value <<"      ";
137         cout <<weight;
138         cout <<"    " <<code <<endl;
139     }
140 }
141
142 };
143
144
145
146 void count_chars( int *counts )
147 {
148     for ( int i = 0 ; i <256 ; i++ )
149         counts[ i ] = 0;
150     ifstream file( "input.dat" );
151     if ( !file ) {
152         cout <<"Couldn't open the input file!\n";
153         throw "abort";
154     }
155     file.setf( ios::skipws );
156     for ( ; ; ) {
157         unsigned char c;
```

```
158         file>> c;
159         if ( file )
160             counts[ c ]++;
161         else
162             break;
163     }
164 }
165
166 int main()
167 {
168     int counts[ 256 ];
169     count_chars( counts );
170     priority_queue < node > q;
171
172     for ( int i = 0 ; i <256 ; i++ )
173         if ( counts[ i ] )
174             q.push( node( i, counts[ i ] ) );
175
176     while ( q.size() >1 ) {
177         node *child0 = new node( q.top() );
178         q.pop();
179         node *child1 = new node( q.top() );
180         q.pop();
181         q.push( node( child0, child1 ) );
182     }
183
184     cout <<"CHAR  FREQUENCY  HOFFMAN-CODE" <<endl;
185     q.top().traverse();
186     return 0;
187 }
```