#include<iostream>

#include<fstream>

#include<string>

using namespace std;

class ctnode

{

public:

int x[9999], info;

char b[9999];

ctnode\* right, \* left;

};

class cnode

{

public:

int x[9999], info;

char a[9999];

cnode\* next;

ctnode\* down;

};

class tree

{

public:

ctnode\* root;

tree()

{

root = NULL;

}

~tree()

{

bye(root);

}

void insert(ctnode\* pnn)

{

if (root == NULL)

{

root = pnn;

}

else

{

if (root->left == NULL)

{

root->left = pnn;

}

else

{

root->right = pnn;

}

}

}

void bye(ctnode\* pt)

{

if (pt == NULL)

{

return;

}

bye(pt->left);

bye(pt->right);

delete pt;

}

void display\_all()

{

display(root);

}

void display(ctnode\* pt)

{

if (pt == NULL)

{

return;

}

display(pt->left);

if (pt->left == NULL && pt->right == NULL)

{

int i = 0;

while (pt->b[i] != '\0')

{

cout << pt->b[i];

i++;

}

cout << ',';

for (int i = 0; pt->x[i] == 0 || pt->x[i] == 1; i++)

{

cout << pt->x[i] << ' ';

}

cout << pt->info << ',';

cout << endl;

}

display(pt->right);

}

void remove(int v)

{

ctnode\* pb = NULL, \* pt = root;

while (pt != NULL && pt->info != v)

{

}

}

};

class CList

{

public:

cnode\* ph, \* pt;

CList()

{

ph = pt = NULL;

}

~CList()

{

cnode\* Ptrav = ph;

while (Ptrav != NULL)

{

ph = ph->next;

delete Ptrav;

Ptrav = ph;

}

}

void sorted\_attach(cnode\* pnn)

{

if (ph == NULL)

{

ph = pnn;

pt = pnn;

}

else

{

cnode\* ptrv = ph, \* pb = ph;

while (ptrv->info < pnn->info)

{

pb = ptrv;

ptrv = ptrv->next;

if (ptrv == NULL)

{

break;

}

}

if (ptrv != pb)

{

pb->next = pnn;

pnn->next = ptrv;

}

else

{

pnn->next = ph;

ph = pnn;

}

}

}

void main\_attach(cnode\* pnn)

{

if (ph == NULL)

{

ph = pt = pnn;

}

else

{

pt->next = pnn;

pt = pnn;

}

}

void disp()

{

cnode\* pt = ph;

while (pt != NULL)

{

int i = 0;

while (pt->a[i] != '\0')

{

cout << pt->a[i];

i++;

}

cout << ',';

for (int i = 0; pt->x[i] == 0 || pt->x[i] == 1; i++)

{

cout << pt->x[i] << ' ';

}

cout << ',';

cout << pt->info;

cout << endl;

pt = pt->next;

}

}

};

void creat\_list(CList& l, char x[], int ct)

{

int k = 0;

char a;

while (k < ct)

{

int ctword = 0;

for (int i = 0; i < ct; i++)

{

if (x[k] == x[i])

{

ctword++;

a = x[i];

}

}

cnode\* ptrv = l.ph;

int tmp = 0;

while (ptrv != NULL)

{

if (ptrv->a[0] == x[k])

{

tmp++;

}

ptrv = ptrv->next;

}

if (tmp == 0)

{

cnode\* pnn = new cnode;

pnn->a[0] = a;

pnn->a[1] = '\0';

pnn->info = ctword;

pnn->next = NULL;

pnn->down = NULL;

l.sorted\_attach(pnn);

}

k++;

}

}

void creat\_tree(CList& l, tree& t)

{

cnode\* ptrv = l.ph, \* ex = ptrv->next;

while (ex != NULL)

{

int i = 0, k = 0;

cnode\* pnn = new cnode;

ctnode\* pnn2 = new ctnode;

while (ptrv->a[i] != '\0')

{

pnn->a[i] = ptrv->a[i];

i++;

}

while (ex->a[k] != '\0')

{

pnn->a[i] = ex->a[k];

i++;

k++;

}

pnn->a[i] = '\0';

pnn->info = ptrv->info + ex->info;

pnn->down = pnn2;

pnn->next = NULL;

i = 0;

while (pnn->a[i] != '\0')

{

pnn2->b[i] = pnn->a[i];

i++;

}

pnn2->info = pnn->info;

pnn2->b[i] = '\0';

pnn2->left = NULL;

pnn2->right = NULL;

t.insert(pnn2);

if (ptrv->down == NULL)

{

ctnode\* left = new ctnode;

i = 0;

while (ptrv->a[i] != '\0')

{

left->b[i] = ptrv->a[i];

i++;

}

left->b[i] = '\0';

left->info = ptrv->info;

left->right = NULL;

left->left = NULL;

t.insert(left);

}

else

{

if (pnn2->left == NULL)

{

pnn2->left = ptrv->down;

}

else

{

pnn2->right = ptrv->down;

}

}

if (ex->down == NULL)

{

ctnode\* right = new ctnode;

k = 0;

while (ex->a[k] != '\0')

{

right->b[k] = ex->a[k];

k++;

}

right->b[k] = '\0';

right->info = ex->info;

right->right = NULL;

right->left = NULL;

t.insert(right);

}

else

{

if (pnn2->right == NULL)

{

pnn2->right = ex->down;

}

else

{

pnn2->left = ex->down;

}

}

l.ph = ex->next;

ex->next = NULL;

delete ptrv;

l.sorted\_attach(pnn);

ptrv = l.ph, ex = ptrv->next;

t.root = NULL;

}

ctnode\* pt = l.ph->down;

t.insert(pt);

}

//turning into list

void Zero\_one\_input(CList& newlist, ctnode\* pt, int\* tmp, int f, int i)

{

if (pt == NULL)

{

i--;

return;

}

if (f == 1)

{

tmp[i] = 1;

i++;

}

if (f == 0)

{

tmp[i] = 0;

i++;

}

Zero\_one\_input(newlist, pt->left, tmp, 0, i);

if (pt->left == NULL && pt->right == NULL)

{

int k = 0;

cnode\* pnn = new cnode;

for (k = 0; k < i; k++)

{

pt->x[k] = tmp[k];

pnn->x[k] = tmp[k];

}

pnn->info = pt->info;

k = 0;

while (pt->b[k] != '\0')

{

pnn->a[k] = pt->b[k];

k++;

}

pnn->a[k] = '\0';

pnn->down = NULL;

pnn->next = NULL;

newlist.main\_attach(pnn);

}

Zero\_one\_input(newlist, pt->right, tmp, 1, i);

}

//counting the leafs

void Zero\_one\_ct(ctnode\* pt, int& zct)

{

if (pt == NULL)

{

return;

}

Zero\_one\_ct(pt->left, zct);

if (pt->left == NULL && pt->right == NULL)

{

zct++;

}

Zero\_one\_ct(pt->right, zct);

}

void compress(CList& l, char x[], char y[], int& t, int& iBit, int ct)

{

char temp = 0;

int k = 0, i = 0;

for (i = 0; i < ct; i++)

{

k = 0;

cnode\* pt = l.ph;

while (pt->a[0] != x[i])

{

pt = pt->next;

}

while (pt->x[k] == 0 || pt->x[k] == 1)

{

if (pt->x[k] == 1)

{

temp = temp | (pt->x[k] << iBit);

}

iBit++;

k++;

y[t] = temp;

if (iBit == 8)

{

iBit = 0;

temp = 0;

t++;

}

}

}

if (iBit != 0 && iBit != 8)

{

t++;

}

}

void text(char\* x)

{

fstream file;

file.open("Debug//input.txt", ios::in | ios::binary);

char Character;

int i = 0;

while (file.read(&Character, 1))

{

x[i] = Character;

i++;

}

}

void txt(int& count)

{

fstream file;

file.open("Debug//input.txt", ios::in | ios::binary);

char Character;

while (file.read(&Character, 1))

{

count++;

}

}

void output(char y[], int ct)

{

fstream file;

file.open("Debug//sympols.txt", ios::out | ios::binary);

int i = 0;

while (i < ct)

{

file << y[i];

i++;

}

}

void Bit(int bit)

{

fstream file;

file.open("Debug//iBit.txt", ios::out | ios::binary);

for (int i = 0; i < bit; i++)

{

file << 1;

}

}

void list\_out(CList& l)

{

fstream file;

file.open("Debug//newlist.txt", ios::out | ios::binary);

cnode\* pt = l.ph;

while (pt != NULL)

{

int i = 0;

file << pt->a[0];

while (pt->x[i] == 0 || pt->x[i] == 1)

{

if (pt->x[i] == 0)

{

file << '0';

}

if (pt->x[i] == 1)

{

file << '1';

}

i++;

}

pt = pt->next;

file << ',';

}

}

void ctimg(int& count)

{

ifstream fl("Debug\\2.bmp", ifstream::binary);

fl.seekg(0, fl.end);

count = fl.tellg();

cout << count << endl;

}

void img(char\* x, int ct)

{

ifstream fl("Debug\\2.bmp", ifstream::binary);

fl.seekg(0, fl.beg);

char ch;

for (int i = 0; i < ct; i++)

{

fl.read(&ch, 1);

x[i] = ch;

//fl2.write(&ch, 1);

//cout << x[i];

}

}

void ct\_vid(int& count)

{

fstream file;

file.open("Debug//khlsana.mp4", ios::in | ios::binary);

char Character;

while (file.read(&Character, 1))

{

count++;

}

}

void video(char\* x)

{

fstream file;

file.open("Debug//khlsana.mp4", ios::in | ios::binary);

char Character;

int i = 0;

while (file.read(&Character,1))

{

x[i] = Character;

i++;

}

}

void main()

{

CList l, newlist;

tree t;

char\* x;

int ct\_read = 0;

//txt(ct\_read);

ctimg(ct\_read);

//ct\_vid(ct\_read);

x = new char[ct\_read];

//text(x);

img(x, ct\_read);

//video(x);

char y[900000];

int c = 0, iBit = 0, zct = 0;

creat\_list(l, x, ct\_read);

creat\_tree(l, t);

ctnode\* pt = t.root;

Zero\_one\_ct(pt, zct);

int\* tmp = new int[zct];

Zero\_one\_input(newlist, pt, tmp, 120, 0);

compress(newlist, x, y, c, iBit, ct\_read);

output(y, c);

list\_out(newlist);

Bit(iBit);

}