```c
/*
** This program implements Optical character Recognition
**.
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
int main()

{
FILE    *fpt1,*fpt2;
unsigned char *image,*template,*norm;
int *MSF;
int mean;
int *temp,max=0,min=0;
int threshold=0;
unsigned char *final;
char    header[320];
char    header2[320];
int   ROWS,COLS,BYTES;
int   ROWS2,COLS2,BYTES2;
int   r,c,r2,c2,sum;
int   r3,c3,r4,c4;
struct timespec tp1,tp2;

  /* read image */
if ((fpt1=fopen("parenthood.ppm","rb")) == NULL)
  {
  printf("Unable to open parenthood.ppm for reading\n");
  exit(0);
  }
if ((fpt2=fopen("parenthood_e_template.ppm","rb")) == NULL)
  {
  printf("Unable to open parenthood_e_template.ppm for reading\n");
  exit(0);
  }
fscanf(fpt1,"%s %d %d %d",header,&COLS,&ROWS,&BYTES);
if (strcmp(header,"P5") != 0  ||  BYTES != 255)
  {
  printf("Not a greyscale 8-bit PPM image\n");
  exit(0);
  }
fscanf(fpt2,"%s %d %d %d",header2,&COLS2,&ROWS2,&BYTES2);
if (strcmp(header2,"P5") != 0  ||  BYTES2 != 255)
  {
  printf("Not a greyscale 8-bit PPM image\n");
  exit(0);
  }
```

```
51    image=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
52    template=(unsigned char *)calloc(ROWS2*COLS2,sizeof(unsigned char));
53    header[0]=fgetc(fpt1);  /* read white-space character that separates header */
54    header[0]=fgetc(fpt2);
55    fread(image,1,COLS*ROWS,fpt1);
56    fclose(fpt1);
57    fread(template,1,COLS2*ROWS2,fpt2);
58    fclose(fpt2);
59      /* allocate memory for final version of image */
60    MSF=(int *)calloc(ROWS2*COLS2,sizeof(int));
61    temp=(int *)calloc(ROWS*COLS,sizeof(int));
62    final=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
63    norm=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));
64
65    sum=0;
66    for (r=0; r<ROWS2; r++)
67      {
68        for (c=0; c<COLS2; c++)
69        {
70        sum+=template[r*COLS2+c];
71        printf("%d\t",template[r*COLS2+c]);
72        }
73      printf("\n");
74      }
75
76        printf("\n\n\n");
77    mean=sum/(ROWS2*COLS2);
78    printf("%d\n",mean);
79    for (r=0; r<ROWS2; r++)
80      {for (c=0; c<COLS2; c++)
81        {
82        MSF[r*COLS2+c]=template[r*COLS2+c]-mean;
83        printf("%d\t",MSF[r*COLS2+c]);
84        }
85      printf("\n");
86      }
87
88    r3=ROWS2/2;
89    c3=COLS2/2;
90    printf("%d %d\n", r3, c3 );
91    for (r=r3; r<ROWS-r3; r++)
92      for (c=c3; c<COLS-c3; c++)
93        {
94        sum=0;
95        for (r2=-r3,r4=0; r2<=r3; r2++,r4++)
96          for (c2=-c3,c4=0; c2<=c3; c2++,c4++)
97            sum+=image[(r+r2)*COLS+(c+c2)] * MSF[r4*COLS2+c4];
98        temp[r*COLS+c]=sum;
99        if(temp[r*COLS+c]>max)
100       {
```

```
101             max=temp[r*COLS+c];
102           }
103           else if(temp[r*COLS+c]<min)
104           {
105             min=temp[r*COLS+c];
106           }
107         }
108         printf("\nmax:%d\tmin:%d",max,min);
109       int max2=0;
110       int min2=0;
111       fpt1=fopen("temp1.ppm","w");
112       fprintf(fpt1,"P5 %d %d 255\n",COLS,ROWS);
113       fwrite(temp,COLS*ROWS,1,fpt1);
114       fclose(fpt1);
115
116       for (r=0; r<ROWS; r++)
117         {for (c=0; c<COLS; c++)
118           {
119             norm[r*COLS+c]= (temp[r*COLS+c]-min)*255/(max-min);
120             if(norm[r*COLS+c]>max2)
121             {
122               max2=norm[r*COLS+c];
123             }
124             else if(norm[r*COLS+c]<min2)
125             {
126               min2=norm[r*COLS+c];
127             }
128           }
129         }
130         printf("\nmax:%d\tmin:%d",max2,min2);
131       fpt1=fopen("temp.ppm","w");
132       fprintf(fpt1,"P5 %d %d 255\n",COLS,ROWS);
133       fwrite(norm,COLS*ROWS,1,fpt1);
134       fclose(fpt1);
135
136       char alphabet;
137       int row,col;
138
139
140       FILE *fpt3;
141
142       for(threshold=0;threshold<=213;threshold++)
143         {
144           for (r=0; r<ROWS; r++)
145         {for (c=0; c<COLS; c++)
146           { if(norm[r*COLS+c]>threshold)
147             final[r*COLS+c]= 255;
148             else
149             final[r*COLS+c]= 0;
150           }
```

```
151     }
152   int gt=0,ob=0;
153   int tp=0,fp=0,fn=0,tn=0;
154   int m=0;
155   int k=0;
156   fpt3 = fopen("parenthood_gt.txt" , "r");
157   r=fscanf(fpt3,"%c %d %d\n",&alphabet,&col,&row);
158   while(m != EOF)
159     {
160       if(alphabet=='e'){
161           gt=1;
162       }
163       else{
164         gt=0;
165       }
166     ob=0;
167
168     for (r=row-r3; r<=row+r3; r++)
169     for (c=col-c3; c<=col+c3; c++)
170       {
171       if(final[r*COLS+c]==255)
172         ob=1; }
173
174       if(gt==1&&ob==1)
175       {tp++;}
176       else if(gt==0&&ob==1)
177       { fp++;}
178       else if(gt==0&&ob==0)
179       { tn++;}
180       else if(gt==1&&ob==0)
181       {fn++;}
182       m=fscanf(fpt3,"%c %d %d\n",&alphabet,&col,&row);
183   }
184   fclose(fpt3);
185   float tpr,fpr;
186   tpr=tp*1.0/((tp+fn)*1.0);
187   fpr=fp*1.0/(fp+tn)*1.0;
188     printf("Threshold: %d tp: %d fp: %d tn: %d fn: %d tpr: %f fpr:
189   %f\n",threshold,tp,fp,tn,fn,tpr,fpr); }
190
191     /* write out final image to see result */
192   fpt1=fopen("final.ppm","w");
193   fprintf(fpt1,"P5 %d %d 255\n",COLS,ROWS);
194   fwrite(final,COLS*ROWS,1,fpt1);
195   fclose(fpt1);
196   }
197
```