

```

1  /*****
2  *
3  * HW04 Q3
4  * Student Name: HASAN MEN
5  * Student ID : 131044009
6  * Date      : 15.3.15
7  * Points    : this program decodes X University's
8  *             messaging system! Assume X University uses a heuristic
9  *             to create encoding table. Their heuristic is just based
10 *             on frequency of letters. Frequently used letters have
11 *             shorter code length.
12 *
13 *****/
14 #include <stdio.h>
15
16 #define TRUE 1
17 #define FALSE 0
18 #define CHARACTERFILE "Files/Q3/CharacterList.txt"
19 #define SAMPLEFILE "Files/Q3/Sample.txt"
20 #define ENCODEDFILE "Files/Q3/XUniversityEncoded.txt"
21 #define PLAINTEXTFILE "Files/Q3/XUniversityMessage.txt"
22
23 /*****
24 * Swaps values of two integers
25 *****/
26 void
27 swap_int(int *a, int *b);
28
29 /*****
30 * Swaps values of two characters
31 *****/
32 void
33 swap_char(char *a, char *b);
34
35 /*****
36 * Sorts characters according to counts. At the end
37 * make sure that *c1 keeps most frequent used letter, *c3
38 * keeps least frequent used letter and *c2 keeps remained
39 * letter
40 *****/
41 void
42 sort(char *a, int a_num, char *b, int b_num, char *c, int c_num);
43
44 /*****
45 * Check whether character is big ASCII letter or not
46 * return TRUE or FALSE
47 *****/
48 int
49 is_letter(char c);
50
51 /*****
52 * Read characters from character list file and if character
53 * is letter assign characters to c1, c2 and c3.
54 * If file has not three letters assign NULL to input char
55 * by order. For ex. file has two letters assign proper
56 * letters to c1 and c2 and assign NULL to c3. If file has
57 * four letters assign c1, c2 and c3 first three letters.
58 * Return number of letters in character list file.
59 * Do not forget to count only proper letters with your
60 * is_letter function. Return number of letters not chars
61 *****/
62 int
63 read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3);
64
65 /*****
66 * Read letters from Sample file and compute frequency of
67 * letters. Then sort it inside this function. Call sort
68 * function. At the end make sure that *c1 keeps most
69 * frequent used letter, *c3 keeps least frequent used
70 * letter and *c2 keeps remained letter
71 *****/
72 void
73 count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3);
74

```

```

75  /*****
76  * Read from XUniversityEncoded file to decode message and *
77  * write decoded (plain text) message to XUniversityMessage *
78  * file. Make sure c1 keeps most frequent used letter, c3 *
79  * keeps least frequent used letter and c2 keeps remained *
80  * letter while calling function. According to frequency *
81  * you know their codes. c1: 0, c2: 10, c3: 110. *
82  *****/
83  void
84  decode(FILE *f_in_ptr, FILE *f_out_ptr, char c1, char c2, char c3);
85
86  /*****
87  * Learns XUniversity's encoding system from given files , *
88  * decodes their encoded messages and writes as plain text to*
89  * a file *
90  *****/
91  int
92  main(int argc, char* argv[])
93  {
94      FILE *f_character_list_ptr, *f_sample_file_ptr, *f_encoded_ptr,
95          *f_plain_text_ptr;
96      char c1, c2, c3;
97
98      /* exit program and print error if files couldn't be opened*/
99      if((f_character_list_ptr=fopen(CHARACTERFILE,"r"))==NULL)
100     {
101         printf("\nCan't open CHARACTERFILE to read\n");
102     }
103     else
104     {
105
106         /* call read_character_list and assign chars to c1,c2,c3 */
107         /* continue if c1,c2,c3 not null */
108         if(read_character_list(f_character_list_ptr, &c1, &c2, &c3)==3)
109         {
110
111             /* close c.list file */
112             fclose(f_character_list_ptr);
113
114
115
116             /* exit program and print error if files couldn't be opened*/
117             if((f_sample_file_ptr=fopen(SAMPLEFILE,"r"))==NULL)
118             {
119                 printf("\nCan't open SAMPLEFILE to read\n");
120             }
121             else
122             {
123
124                 /* call count letter func. */
125                 count_letters(f_sample_file_ptr,&c1,&c2,&c3);
126
127                 /* close sample file*/
128                 fclose(f_sample_file_ptr);
129
130                 /* exit program and print error if files couldn't be opened*/
131                 if((f_encoded_ptr=fopen(ENCODEDFILE,"r"))==NULL ||
132                     (f_plain_text_ptr=fopen(PLAINTEXTFILE,"w"))==NULL )
133                 {
134                     if((f_encoded_ptr=fopen(ENCODEDFILE,"r"))==NULL)
135                         printf("Can't open ENCODEDFILE to read");
136                     else printf("Can't open PLAINTEXTFILE to write");
137                 }
138
139                 /*files opened */
140                 else
141                 {
142
143                     /* call decode function and write new message */
144                     decode(f_encoded_ptr,f_plain_text_ptr,c1,c2,c3);
145
146                     /* close files */
147                     fclose(f_encoded_ptr);
148                     fclose(f_plain_text_ptr);

```

```

149         }
150     }
151 }
152     else printf("\nNumber of letter not equal 3 - Program finished\n");
153 }
154
155     printf("****    Mission Completed    ****\n");
156
157     return 0;
158 }
159
160 /*****
161  * Swaps values of two integers
162  *****/
163 void
164 swap_int(int *a, int *b)
165 {
166     int term;
167     term = *a;
168     *a=*b;
169     *b=term;
170 }
171
172 /*****
173  * Swaps values of two characters
174  *****/
175 void
176 swap_char(char *a, char *b)
177 {
178     char term;
179     term = *a;
180     *a=*b;
181     *b=term;
182 }
183
184 /*****
185  * Sorts characters according to counts. At the end
186  * make sure that *c1 keeps most frequent used letter, *c3
187  * keeps least frequent used letter and *c2 keeps remained
188  * letter
189  *****/
190 void
191 sort(char *a, int a_num, char *b, int b_num, char *c, int c_num)
192 {
193
194     /* sort c1,c2,c3 accoring to most frequent used */
195     if(a_num<=b_num)
196     {
197         swap_char(a,b);
198         swap_int(&a_num,&b_num);
199     }
200     if(a_num<=c_num)
201     {
202         swap_char(a,c);
203         swap_int(&a_num,&c_num);
204     }
205
206     if(b_num<=c_num)
207     {
208         swap_char(b,c);
209         swap_int(&b_num,&c_num);
210     }
211
212     /* printf("%c=%d %c=%d %c=%d\n",*a,a_num,*b,b_num,*c,c_num); */
213     /* check c1,c2,c3 and thehir used numbers after shorting*/
214 }
215
216 /*****
217  * Check whether character is big ASCII letter or not
218  * return TRUE or FALSE
219  *****/
220 int
221 is_letter(char c)
222 {

```

```

223     if(c<='Z' && c >='A')
224         return TRUE;
225     else
226         return FALSE;
227 }
228
229 /*****
230  * Read characters from character list file and if character *
231  * is letter assign characters to c1, c2 and c3. *
232  * If file has not three letters assign NULL to input char *
233  * by order. For ex. file has two letters assign proper *
234  * letters to c1 and c2 and assign NULL to c3. If file has *
235  * four letters assign c1, c2 and c3 first three letters. *
236  * Return number of letters in character list file. *
237  * Do not forget to count only proper letters with your *
238  * is_letter function. Return number of letters not chars *
239  *****/
240 int
241 read_character_list(FILE* f_in_ptr, char *c1, char *c2, char *c3)
242 {
243     int counter = 0; /*local variable, counts proper letter */
244     char character; /* keeps character from file*/
245     while((fscanf(f_in_ptr, " %c",&character))!=EOF)
246     {
247         if(is_letter(character)==TRUE)
248         {
249             counter++;
250
251             if(counter==1)
252             {
253                 *c1=character;
254                 *c2=0; /* c2=null*/
255             }
256             else if(counter==2)
257             {
258                 *c2=character;
259                 *c3=0; /* c3=null*/
260             }
261             else if(counter==3)
262             {
263                 *c3=character;
264             }
265         }
266     }
267     return counter;
268 }
269
270 /*****
271  * Read letters from Sample file and compute frequency of *
272  * letters. Then sort it inside this function. Call sort *
273  * function. At the end make sure that *c1 keeps most *
274  * frequent used letter, *c3 keeps least frequent used *
275  * letter and *c2 keeps remained letter *
276  *****/
277 void
278 count_letters(FILE *f_in_ptr, char *c1, char *c2, char *c3)
279 {
280     char letter; /* keeps character from sample file*/
281     int a_num=0,b_num=0,c_num=0;
282
283     while(fscanf(f_in_ptr, " %c",&letter)!=EOF)
284     {
285         if(letter==*c1)
286         {
287             a_num++;
288         }
289         else if(letter==*c2)
290         {
291             b_num++;
292         }
293         else if(letter==*c3)
294         {
295             c_num++;
296         }

```

```

297     }
298     /*check c1,c2,c3 and their number before shorting */
299     /*printf("%c=%d %c=%d %c=%d\n",*c1,a_num,*c2,b_num,*c3,c_num);*/
300     sort(c1,a_num,c2,b_num,c3,c_num);
301 }
302
303 /*****
304  * Read from XUniversityEncoded file to decode message and *
305  * write decoded (plain text) message to XUniversityMessage *
306  * file. Make sure c1 keeps most frequent used letter, c3 *
307  * keeps least frequent used letter and c2 keeps remained *
308  * letter while calling function. According to frequency *
309  * you know their codes. c1: 0, c2: 10, c3: 110. *
310  *****/
311 void
312 decode(FILE *f_in_ptr, FILE *f_out_ptr, char c1, char c2, char c3)
313 {
314     char character;
315     int number_of_ones=0;
316     while(fscanf(f_in_ptr, " %c",&character)!=EOF)
317     {
318
319         /* count number of ones until take '0' and write c1,c2,c3 according
320          * known codes */
321         if(character=='1')
322         {
323             number_of_ones++;
324         }
325         else if(character=='0')
326         {
327             switch(number_of_ones)
328             {
329                 case 0: fprintf(f_out_ptr,"%c",c1); break;
330                 case 1: fprintf(f_out_ptr,"%c",c2); break;
331                 case 2: fprintf(f_out_ptr,"%c",c3); break;
332             }
333             number_of_ones=0; /* rest the number of ones */
334         }
335     }
336 }
337
338 /* end of HW04_HASAN_MEN_131044009_part3.c */

```