

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<math.h>
4
5
6  int main(){
7
8
9  int op;
10 char secretCode='A';//has a default value of 'A'
11 //these b1-b8 represents an 8 bit integer. It is equivalent the binary equivalent of 'A',which is 65
12
13 int b8=0,b7=1,b6=0,b5=0,b4=0,b3=0,b2=0,b1=1;//these values can be modified If the user selects option1 and
changes the secret code.
14 int ecB8=0,ecB7=0,ecB6=0,ecB5=0,ecB4=0,ecB3=0,ecB2=0,ecB1=0;//ecB stands for "encrypted bit"
15 int hexToB8=0,hexToB7=1,hexToB6=0,hexToB5=0,hexToB4=0,hexToB3=0,hexToB2=0,hexToB1=1;//hexToB stands for
"hexadecimal to binary bit. This is used in option2-base16"
16 int base;//this variable used in the 2nd option as a base choice.
17
18
19 do{
20     printf("\nHappy Encryption!\n");
21     printf("(1)Change the secret code\n");
22     printf("(2)Encrypt text\n");
23     printf("(3)Quit\n");
24     printf("You choose: ");
25     scanf("%d",&op);
26     while(op<1 || op>3){
27         fflush(stdin);
28         printf("Please enter an option between 1-3!\n");
29         printf("You choose: ");
30         scanf("%d",&op);
31     }
32
33     switch(op){
34
35     case 1:
36
37         fflush(stdin);//clean the buffer before requesting for character input
38         printf("You have chosen option 1!\n");
39         printf("Which secret code will you use? ");//keep in mind that you must only enter a single character
40         scanf("%c",&secretCode);
41         while((secretCode<48||secretCode>57) && (secretCode<65||secretCode>90) && (secretCode<97||secretCode>
122)){
42             fflush(stdin);//clean the buffer before requesting for character input
43             printf("\nERROR! Secret code can be either:\n");
44             printf("A digit between 0-9\n");
45             printf("Uppercase letters\n");
46             printf("Lowercase characters\n\n");
47             printf("Which secret code will you use? ");
48             scanf("%c",&secretCode);
49         }
50
51         int rem;//this variable is the remainder when we take modulus 2 of the number
52         int number=secretCode;//this hold the integer value of the secret code
53         int binaryNumber=0;// this is our final conversion result
54         int mult=1;//this is for multiplier
55         int count=0;//this is the count of binary number elements
56
57         while(number>0){
58             rem=number%2;
59             binaryNumber=binaryNumber+(rem*mult);
60             mult*=10;
61             number/=2;
62             count++;
63         }

```

```

64
65 //as we cant use arrays, I will modify every bit till the number becomes zero
66
67 //this means that the bit8 must remains zero, and I need to modify the bit1 to bit7
68 if(count==7){
69
70     b1=binaryNumber%10;
71     binaryNumber/=10;
72     b2=binaryNumber%10;
73     binaryNumber/=10;
74     b3=binaryNumber%10;
75     binaryNumber/=10;
76     b4=binaryNumber%10;
77     binaryNumber/=10;
78     b5=binaryNumber%10;
79     binaryNumber/=10;
80     b6=binaryNumber%10;
81     binaryNumber/=10;
82     b7=binaryNumber%10;
83     binaryNumber/=10;
84     b8=0;
85 }
86 //this means that the bit7 and 8 must remains zero, and I need to modify the bit1 to bit6
87 else{
88
89     b1=binaryNumber%10;
90     binaryNumber/=10;
91     b2=binaryNumber%10;
92     binaryNumber/=10;
93     b3=binaryNumber%10;
94     binaryNumber/=10;
95     b4=binaryNumber%10;
96     binaryNumber/=10;
97     b5=binaryNumber%10;
98     binaryNumber/=10;
99     b6=binaryNumber%10;
100    binaryNumber/=10;
101    b7=0;
102    b8=0;
103
104 }
105
106 //printing the binary equivalent of the secret code.
107 printf("Binary equivalent of the chosen code is ");
108 printf("%d%d%d%d%d%d%d",b8,b7,b6,b5,b4,b3,b2,b1);
109 break;
110
111 case 2:
112
113     fflush(stdin);
114     printf("You have chosen option 2!\n");
115     printf("Which base will you use to enter text (base 16/2)? ");
116     scanf("%d",&base);
117     while(base!=2 && base!=16){
118         fflush(stdin);
119         printf("Please enter base either 2 or 16! ");
120         scanf("%d",&base);
121     }
122
123     if(base==2){
124         char ch;
125         int errorFlag=0,errorBinaryFlag=0;//if error flag equals 1, then user need to enter the text again.
126         int asciiValue,lengthCount=0;//length is initialized as -1 as I dont count the enter as a bit.
127
128         while(!errorFlag){

```

```

130     printf("Please enter the text to encrypt: ");
131     fflush(stdin);
132     do{
133
134         scanf("%c",&ch);
135         asciiValue=ch;
136
137
138         if(asciiValue!=48&&asciiValue!=49&&asciiValue!=10){//checking whether entered character is
either 0 or 1.
139
140             errorBinaryFlag=1;
141             break;
142
143         }
144         if(asciiValue!=10){// I will not count the enter as a character
145
146             lengthCount++;
147
148             //applying XOR operation to every bit.
149             if(lengthCount%8==0){
150                 if(b1!=asciiValue-48){
151
152                     ecB1=1;
153                 }
154                 else{
155
156                     ecB1=0;
157                 }
158                 int encodedText;//this is the decimal value of encoded Text.
159                 encodedText=ecB8*pow(2,7)+ecB7*pow(2,6)+ecB6*pow(2,5)+ecB5*pow(2,4)+ecB4*pow(2,3)+
ecB3*pow(2,2)+ecB2*pow(2,1)+ecB1*pow(2,0);
160
161                 printf("%c",encodedText);//printing the encoded text
162             }
163             else if(lengthCount%8==7){
164                 if(b2!=asciiValue-48){
165
166                     ecB2=1;
167                 }
168                 else{
169
170                     ecB2=0;
171                 }
172             }
173             else if(lengthCount%8==6){
174                 if(b3!=asciiValue-48){
175
176                     ecB3=1;
177                 }
178                 else{
179                     ecB3=0;
180                 }
181             }
182             else if(lengthCount%8==5){
183                 if(b4!=asciiValue-48){
184
185                     ecB4=1;
186                 }
187                 else{
188                     ecB4=0;
189                 }
190             }
191             else if(lengthCount%8==4){
192                 if(b5!=asciiValue-48){

```

```

194         ecB5=1;
195     }
196     else{
197         ecB5=0;
198     }
199 }
200 else if(lengthCount%8==3){
201     if(b6!=asciiValue-48){
202
203         ecB6=1;
204     }
205     else{
206         ecB6=0;
207     }
208 }
209 else if(lengthCount%8==2){
210     if(b7!=asciiValue-48){
211
212         ecB7=1;
213     }
214     else{
215         ecB7=0;
216     }
217 }
218 else if(lengthCount%8==1){
219     if(b8!=asciiValue-48){
220
221         ecB8=1;
222     }
223     else{
224         ecB8=0;
225     }
226 }
227
228 }
229 }
230 }
231
232
233
234
235 }while(ch!=10);
236
237 //these are error conditions, If there is any, the loop will continue.
238 if(errorBinaryFlag==1){
239     printf("Please enter only digits 0 and 1! ");
240     errorBinaryFlag=0;
241     lengthCount=0;
242     continue;
243 }
244 else{
245     if(lengthCount<8){
246         printf("Please enter at least 8 bits for the text! ");
247         lengthCount=0;
248         continue;
249     }
250     else{
251         errorFlag=1;
252     }
253 }
254 }
255
256 }
257
258
259

```

```

260
261 }
262 else if(base==16){
263     char ch;
264     int errorFlag=0,errorHexFlag=0;
265     int asciiValue,lengthCount=0;
266
267     printf("Please enter the text to encrypt: ");
268     while(!errorFlag){
269
270
271         fflush(stdin);
272         do{
273
274             scanf("%c",&ch);
275             asciiValue=ch;
276             //the entered character must be between 0-9 or A-F
277             if((asciiValue<48||asciiValue>57)&&(asciiValue<65||asciiValue>70)&&(asciiValue!=10)){
278
279                 errorHexFlag=1;
280                 break;
281
282             }
283
284             if(asciiValue!=10){
285                 lengthCount++;
286             }
287
288             //here, I declared 0 or 1 to every bit based on the entered character from the user
289             if(lengthCount%2==1){
290                 switch(ch){
291                     case '0':
292                         hexToB8=0;
293                         hexToB7=0;
294                         hexToB6=0;
295                         hexToB5=0;
296                         break;
297                     case '1':
298                         hexToB8=0;
299                         hexToB7=0;
300                         hexToB6=0;
301                         hexToB5=1;
302                         break;
303                     case '2':
304                         hexToB8=0;
305                         hexToB7=0;
306                         hexToB6=1;
307                         hexToB5=0;
308                         break;
309                     case '3':
310                         hexToB8=0;
311                         hexToB7=0;
312                         hexToB6=1;
313                         hexToB5=1;
314                         break;
315                     case '4':
316                         hexToB8=0;
317                         hexToB7=1;
318                         hexToB6=0;
319                         hexToB5=0;
320                         break;
321                     case '5':
322                         hexToB8=0;
323                         hexToB7=1;
324                         hexToB6=0;
325                         hexToB5=1;

```

```

326         break;
327     case '6':
328         hexToB8=0;
329         hexToB7=1;
330         hexToB6=1;
331         hexToB5=0;
332         break;
333     case '7':
334         hexToB8=0;
335         hexToB7=1;
336         hexToB6=1;
337         hexToB5=1;
338         break;
339     case '8':
340         hexToB8=1;
341         hexToB7=0;
342         hexToB6=0;
343         hexToB5=0;
344         break;
345     case '9':
346         hexToB8=1;
347         hexToB7=0;
348         hexToB6=0;
349         hexToB5=1;
350         break;
351     case 'A':
352         hexToB8=1;
353         hexToB7=0;
354         hexToB6=1;
355         hexToB5=0;
356         break;
357     case 'B':
358         hexToB8=1;
359         hexToB7=0;
360         hexToB6=1;
361         hexToB5=1;
362         break;
363     case 'C':
364         hexToB8=1;
365         hexToB7=1;
366         hexToB6=0;
367         hexToB5=0;
368         break;
369     case 'D':
370         hexToB8=1;
371         hexToB7=1;
372         hexToB6=0;
373         hexToB5=1;
374         break;
375     case 'E':
376         hexToB8=1;
377         hexToB7=1;
378         hexToB6=1;
379         hexToB5=0;
380         break;
381     case 'F':
382         hexToB8=1;
383         hexToB7=1;
384         hexToB6=1;
385         hexToB5=1;
386         break;
387
388     }
389 }
390 else{
391     switch(ch){

```

```
392     case '0':
393         hexToB4=0;
394         hexToB3=0;
395         hexToB2=0;
396         hexToB1=0;
397         break;
398     case '1':
399         hexToB4=0;
400         hexToB3=0;
401         hexToB2=0;
402         hexToB1=1;
403         break;
404     case '2':
405         hexToB4=0;
406         hexToB3=0;
407         hexToB2=1;
408         hexToB1=0;
409         break;
410     case '3':
411
412         hexToB4=0;
413         hexToB3=0;
414         hexToB2=1;
415         hexToB1=1;
416         break;
417     case '4':
418         hexToB4=0;
419         hexToB3=1;
420         hexToB2=0;
421         hexToB1=0;
422         break;
423     case '5':
424         hexToB4=0;
425         hexToB3=1;
426         hexToB2=0;
427         hexToB1=1;
428         break;
429     case '6':
430         hexToB4=0;
431         hexToB3=1;
432         hexToB2=1;
433         hexToB1=0;
434         break;
435     case '7':
436         hexToB4=0;
437         hexToB3=1;
438         hexToB2=1;
439         hexToB1=1;
440         break;
441     case '8':
442         hexToB4=1;
443         hexToB3=0;
444         hexToB2=0;
445         hexToB1=0;
446         break;
447     case '9':
448         hexToB4=1;
449         hexToB3=0;
450         hexToB2=0;
451         hexToB1=1;
452         break;
453     case 'A':
454         hexToB4=1;
455         hexToB3=0;
456         hexToB2=1;
457         hexToB1=0;
```

```

458         break;
459     case 'B':
460         hexToB4=1;
461         hexToB3=0;
462         hexToB2=1;
463         hexToB1=1;
464         break;
465     case 'C':
466         hexToB4=1;
467         hexToB3=1;
468         hexToB2=0;
469         hexToB1=0;
470         break;
471     case 'D':
472         hexToB4=1;
473         hexToB3=1;
474         hexToB2=0;
475         hexToB1=1;
476         break;
477     case 'E':
478
479         hexToB4=1;
480         hexToB3=1;
481         hexToB2=1;
482         hexToB1=0;
483         break;
484     case 'F':
485         hexToB4=1;
486         hexToB3=1;
487         hexToB2=1;
488         hexToB1=1;
489         break;
490     }
491     if(ch!=10){
492
493
494         //If no problem, I will declared encrypted text bits.
495         if(b1!=hexToB1){
496
497             ecB1=1;
498         }
499         else{
500
501             ecB1=0;
502         }
503
504
505
506
507
508         if(b2!=hexToB2){
509
510             ecB2=1;
511         }
512         else{
513
514             ecB2=0;
515         }
516
517
518         if(b3!=hexToB3){
519
520             ecB3=1;
521         }
522         else{
523             ecB3=0;

```



```

524     }
525
526     if(b4!=hexToB4){
527
528         ecB4=1;
529     }
530     else{
531         ecB4=0;
532     }
533
534
535
536     if(b5!=hexToB5){
537
538         ecB5=1;
539     }
540     else{
541         ecB5=0;
542     }
543
544
545     if(b6!=hexToB6){
546
547         ecB6=1;
548     }
549     else{
550         ecB6=0;
551     }
552
553
554     if(b7!=hexToB7){
555
556         ecB7=1;
557     }
558     else{
559         ecB7=0;
560     }
561
562
563     if(b8!=hexToB8){
564
565         ecB8=1;
566     }
567     else{
568         ecB8=0;
569     }
570
571     int encodedText;
572     encodedText=ecB8*pow(2,7)+ecB7*pow(2,6)+ecB6*pow(2,5)+ecB5*pow(2,4)+ecB4*pow(2,3)+
ecB3*pow(2,2)+ecB2*pow(2,1)+ecB1*pow(2,0);
573     //printing encoded text
574     printf("%c",encodedText);
575
576     }
577
578
579     }
580 }while(ch!=10);
581
582
583 //these are error conditions, If there is any, the loop will continue.
584 if(errorHexFlag==1){
585     printf("Please enter only 0-9 or A-f ");
586     errorHexFlag=0;
587     lengthCount=0;
588     continue;

```

```
589     }
590     else{
591         if(lengthCount<2){
592             printf("Please enter at least 2 bits for the text! ");
593             lengthCount=0;
594             continue;
595         }
596         else{
597             errorFlag=1;
598         }
599     }
600 }
601 }
602 }
603 }
604 }
605 }
606 break;
607 }
608 case 3:
609     printf("See You!\n");
610     break;
611 }
612 }
613 }
614 }
615 }
616 }while(op!=3);
617 }
618 }
619 }
620 return 0;
621 }
622 }
```