**EEL 2880: Spr 2020: 03\_12\_2020: Dynamic Coding and Data Security**

**Deitel/Subbarao: Reference: Chapters 6, 7, 8, and 9 and 10**

**Copy Right extended through 2020**

**Introduction:** In the present times, with all the information transfer on Internet and wireless, data security is being compromised. All the institutions, governments, bio and other industries are heavily concerned about the data and individual security. While there is nothing absolute security, several methods are being devised to protect the data and information security. Some of the methodologies are: Data Encryption Schemes [DES], where mathematical process takes the base information through iterations to modify the content; Stegenography Schemes, where in the digital scenario, some bits of information are randomly replaced by the data elements and mask the information in a big digital platform; Dynamic codining Schemes, where the number codes of characters are dynamically changed and the modified information is being communicated; and several such schemes. In all these methodologies, the reverse key should be available for the proper individuals to be able to decode and obtain the original message. To protect the nature of the data security, whatever methodology is being followed, should be made dynamic, so that it would not be broken.

Data security is becoming increasingly demanding activity in all the fileds of human existance. Here is a sample of dynamic coding scheme:

// Data Security: Dynamic coding: Chaps 6-9 and 10

 // Deitel/Subbarao EEL 2880: Modified 10 02 2018

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

int main()

{

 char s[ 100 ]; // define character array of size 100

 size\_t i; // loop counter

 puts( "\n\a Data Security: Chps 6-9; Dynamic coding:\n " );

 puts( "\n\a Deitel/Subbarao EEL 2880 F 2014 10 21 2014 \n " );

 // use gets to get text from user

 puts( "Enter a line of text: \a\n" );

 fgets( s, 100, stdin );

 puts( "\n\aThe line in uppercase is: \n" );

 // convert each character to ASCII uppercase and output

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( "%c", toupper( s[ i ]) );

 } // end for Uppercase characters

 puts("\n upper case ASCII code for the characters \n");

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf(" %c:%d",toupper(s[ i ]),toupper(s[ i ]));

 } // end for Uppercase ASCII

 // convert each character to modified ASCII uppercase and output

 puts( "\n\aThe line in modified uppercase with progressively higher ASCII:\n" );

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( "%c", toupper( s[ i ] + (i+1)) );

 } // end for modified Uppercase characters

 puts("");

 printf( " \n Uppercase modified ASCII code for the entered string \n\n\a");

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( " %c:%d",toupper(s[ i ] + (i+1)),toupper(s[ i ] +(i+1)));

 } // end for modified Uppercase ASCII code

 printf( " \n\n Lower case for the entered string \n\n\a");

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( "%c",tolower (s[ i ]));

 } // end for Lowercase ASCII

 printf( " \n Lower case ASCII for the entered string \n\n\a");

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( " %c:%d",tolower( s[ i ]),tolower( s[ i ]));

 } // end for Lowercase ASCII

printf( " \n modified Lower case ASCII for the entered string \n\n\a");

for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( " %c:%d", tolower ( s[ i ] - (i+1)),tolower( s[ i ] - (i+1)));

 } // end for modified Lowercase ASCII

 // convert each character to modified ASCII uppercase and output

 puts( "\n\aThe line in modified lowercase with progressively lower ASCII:\n" );

 for ( i = 0; s[ i ] != '\0'; ++i )

 {

 printf( "%c", tolower( s[ i ] - (i+1)) );

 } // end for modified lower characters

 puts("");

 printf( "\n \a\t The final value of loop variable 'i' is:%d\n", i);

 puts( "" );

 } // end main



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 0123456789101112131415 | 0123456789ABCDEF | NULSOHSTXETXEOTENQACKBELBSTABLFVTFFCRSOSI |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 16171819202122232425262728293031 | 101112131415161718191A1B1C1D1E1F | DLEDC1DC2DC3DC4NAKSYNETBCANEMSUBESCFSGSRSUS |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 32333435363738394041424344454647 | 202122232425262728292A2B2C2D2E2F | (space)!"#$%&'()\*+,-./ |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 48495051525354555657585960616263 | 303132333435363738393A3B3C3D3E3F | 0123456789:;<=>? |

 |
|

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 64656667686970717273747576777879 | 404142434445464748494A4B4C4D4E4F | @ABCDEFGHIJKLMNO |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 80818283848586878889909192939495 | 505152535455565758595A5B5C5D5E5F | PQRSTUVWXYZ[\]^\_ |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 96979899100101102103104105106107108109110111 | 606162636465666768696A6B6C6D6E6F | `abcdefghijklmno |

 |

|  |  |  |
| --- | --- | --- |
| **ASCII** | **Hex** | **Symbol** |
|  |
| 112113114115116117118119120121122123124125126127 | 707172737475767778797A7B7C7D7E7F | pqrstuvwxyz{|}~ |

 |