```c
/*****************************************************************************
*   FILE: affine.c
*   AUTHOR: Connor Beardsmore - 15504319
*   UNIT: FCC200
*   PURPOSE: Run Affine cipher given text and key, either encrypt or decrypt
*   LAST MOD: 28/03/17
*   REQUIRES: affine.h
*****************************************************************************/

#include "affine.h"

//----------------------------------------------------------------------------
// FUNCTION: main

int main( int argc, char* argv[] )
{
    if ( argc != ARGS )
    {
        printf("\nUSAGE: <FLAG> <INPUT FILE> <OUTPUT FILE> <KEY A> <KEY B>\n");
        printf("FLAGS ARE: -e for encryption, -d for decryption\n\n");
        return 1;
    }

    // CONVERT ARGC NAMES
    char* flag = argv[1];
    char* inFile = argv[2];
    char* outFile = argv[3];
    int a = atoi( argv[4] );
    int b = atoi( argv[5] );

    // CHECK THAT THE KEYS ARE ELIGIBLE
    int validity = keyEligible( a, b, ALPHABET );
    if ( validity != 1 )
    {
        printf("\nKEYS %d AND %d ARE NOT VALID.\n", a, b );
        return 2;
    }

    // OPEN INPUT AND OUTPUT FILES
    FILE* inF = fopen( inFile, "r" );
    FILE* outF = fopen( outFile, "w" );

    // CHECK OPEN FOR ERRORS
    if ( ( inF == NULL ) || ( outF == NULL ) )
    {
        perror("\nERROR OPENING INPUT OR OUTPUT FILE\n");
        return 3;
    }

    // FUNCTION POINTER FOR encrypt() OR decrypt()
    FuncPtr fp;

    // PERFORM ENCRYPTION IF -e FLAG PROVIDED AND VICE VERSA
    if ( !strncmp( flag, "-e", 2 ) )
        fp = &encrypt;
    else if ( !strncmp( flag, "-d", 2 ) )
        fp = &decrypt;
    else
    {
        printf("\nFLAG IS INCORRECT, MUST BE -e OR -d\n");
        return 4;
    }

    // PERFROM APPROPRIATE FUNCTION
    while ( ( feof( inF ) == 0 ) && ( ferror( inF ) == 0) && (ferror( inF ) == 0) )
    {
        // GET THE NEXT CHARACTER FROM FILE
        char next = fgetc( inF );
        if ( feof( inF ) == 0 )
            // WRITE THE CONVERTED CHARACTER TO FILE
            fputc( ( *fp )( next, a, b ), outF );
    }
```

```
73
74      // CLOSE FILES
75      fclose( inF );
76      fclose( outF );
77
78      return 0;
79  }
80
81  //-------------------------------------------------------------------------
82  // FUNCTION: encrypt
83  // IMPORT: plain (char), a (int), b (int)
84  // PURPOSE: Convert a plaintext char into the encryped character
85
86  char encrypt( char plain, int a, int b)
87  {
88      char output = plain;
89      // ENCRYPT BASED ON plain * a + b MODULO 26
90      // IGNORE NON-CHARACTERS
91      if ( isupper(plain) )
92          output = ( ( ( plain - 'A' ) * a + b ) % ALPHABET ) + 'A';
93      else if ( islower(plain) )
94          output = ( ( ( plain - 'a' ) * a + b ) % ALPHABET ) + 'a';
95      return output;
96  }
97
98  //-------------------------------------------------------------------------
99  // FUNCTION: decrypt
100 // IMPORT: plain (char*), a (int), b (int)
101 // PURPOSE: Convert a ciphertect char into the decrypted character
102
103 char decrypt( char cipher, int a, int b )
104 {
105     // FIND THE MODULO INVERSE USING EUCLIDEAN
106     int inverse = extendEuclid( a, ALPHABET );
107     char output = cipher;
108     // DECRYPT BASED ON inverse * cipher - b MODULO 26
109     // IGNORE NON-CHARACTERS
110     if ( isupper(cipher) )
111         output = ( ( inverse * ( cipher - 'A' - b + ALPHABET ) ) % ALPHABET ) + 'A';
112     else if ( islower(cipher) )
113         output = ( ( inverse * ( cipher - 'a' - b + ALPHABET ) ) % ALPHABET ) + 'a';
114     return output;
115 }
116
117 //-------------------------------------------------------------------------
118
```