```c
/***************************
 * Problem Set 2           *
 * caesar.c                *
 *                         *
 * Doug Lloyd              *
 * September 21, 2011      *
 ***************************/

/* Constants */
#define ALPHASIZE 26
#define LOWERBASE 'a'
#define UPPERBASE 'A'

/* Header Files */
#include <stdio.h>
#include <ctype.h>
#include <cs50.h>
#include <string.h>
#include <stdlib.h>

/* Function Declarations */
char caesarshift(char c, int shift);


/****************************/
int main(int argc, char *argv[]) {

  /* Ensure proper number of command line arguments */
  if(argc != 2) {
    printf("Usage: ./caesar <shift>\n");
    return 1;
  }

  /* Convert shift to useable form */
  int shift = atoi(argv[1]);

  /* Prompt user for string to encipher */
  printf("Plaintext: ");
  string ptxt = GetString();

  /* Execute Caesar shift on plaintext string, one character at a time */
  for(int i = 0, len = strlen(ptxt); i < len; i++)
    printf("%c", caesarshift(ptxt[i], shift));

  /* Completed everything, print newline and exit */
  printf("\n");
  return 0;
}

char caesarshift(char c, int shift) {
  /* Change character from ASCII value to range [0-25], add the shift,
     account for wrap around, and calculate new ASCII value */
  if(islower(c))
    return ((c - LOWERBASE) + shift) % ALPHASIZE + LOWERBASE;
  else if(isupper(c))
    return ((c - UPPERBASE) + shift) % ALPHASIZE + UPPERBASE;

  /* Non alphabetic characters are unshifted */
  else
    return c;
}
```

```c
/****************************
 * Problem Set 2            *
 * oldman.c                 *
 *                          *
 * Doug Lloyd               *
 * September 21, 2011       *
 ****************************/

/* Constants */
#define VERSES 10

/* Header Files */
#include <stdio.h>
#include <cs50.h>

/* Function Declarations */
string where(int versenum);

/*****************************/

int main(int argc, char *argv[]) {

  /* Arrays to hold strings for verse number and locations */
  string nums[VERSES] = {"one", "two", "three", "four", "five",
                         "six", "seven", "eight", "nine", "ten"};
  string places[VERSES] = {"thumb", "shoe", "knee", "door", "hive",
                           "sticks", "heaven", "gate", "spine", "again"};

  /* Singing the song */
  for(int i = 0; i < VERSES; i++) {
    printf("This old man, he played %s\n", nums[i]);
    printf("He played knick-knack %s %s\n", where(i), places[i]);
    printf("Knick-knack, paddywhack, give your dog a bone\n");
    printf("This old man came rolling home\n\n");
  }

  return 0;
}

string where(int versenum) {

  /* Depending on the verse number, the sentence changes a little bit */
  switch(versenum) {
  case 6:
    return "up in";
    break;
  case 9:
    return "once";
    break;
  default:
    return "on my";
    break;
  }

  /* The program should never get here. */
  return "";
}
```

```c
/***************************
 * Problem Set 2           *
 * vigenere.c              *
 *                         *
 * Doug Lloyd              *
 * September 21, 2011      *
 ***************************/

/* Constants */
#define ALPHASIZE 26
#define LOWERBASE 'a'
#define UPPERBASE 'A'

/* Header Files */
#include <stdio.h>
#include <ctype.h>
#include <cs50.h>
#include <string.h>
#include <stdlib.h>

/* Function Declarations */
char vigshift(char c, char shift);


/****************************/
int main(int argc, char *argv[]) {

  /* Ensure proper number of command line arguments */
  if(argc != 2) {
    printf("Usage: ./vigenere <keyword>\n");
    return 1;
  }

  /* Iterate over characters in keyword, ensure all alphabetic */
  int klen = strlen(argv[1]);

  for(int i = 0; i < klen; i++)
    if(!isalpha(argv[1][i])) {
      printf("Keyword must contain only alphabetic characters.\n");
      return 2;
    }

  /* Prompt user for plaintext to encipher */
  printf("Plaintext: ");
  string ptxt = GetString();

  /* Loop through, enciphering one character at a time */
  for(int i = 0, kpos = 0, len = strlen(ptxt); i < len; i++) {

    /* If we reached end of keyword, go back to the beginning */
    if(kpos == klen)
      kpos = 0;

    /* We encipher alphabetic characters and advance our keyword position */
    if(isalpha(ptxt[i])) {
      printf("%c", vigshift(ptxt[i], argv[1][kpos]));
      kpos++;
    }

    /* And we leave the other ones alone */
    else
      printf("%c", ptxt[i]);
  }

  /* All done! */
  printf("\n");
  return 0;
}

char vigshift(char c, char shift) {

  /* Reduce the shifting letter from ASCII to the [0-25] range */
  if(islower(shift))
    shift -= LOWERBASE;
  else
    shift -= UPPERBASE;

  /* Execute the shift: Reduce plaintext character to range [0-25], add
     shift value, account for wrap around, and return to ASCII */
  if(islower(c))
    return ((c - LOWERBASE) + shift) % ALPHASIZE + LOWERBASE;
  else
    return ((c - UPPERBASE) + shift) % ALPHASIZE + UPPERBASE;
}
```