```c
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Prototype declaration
bool only_digits(string s);
char rotate(char c, int n);

int main(int argc, string argv[])
{
    // Check for a single command-line argument
    if (argc != 2)
    {
        printf("Usage: ./caesar key\n");
        return 1;
    }

    // Verify the command-line argument is all digits
    if (!only_digits(argv[1]))
    {
        printf("Usage: ./caesar key\n");
        return 1;
    }

    // Convert argv[1] from a `string` to an `int`
    int key = atoi(argv[1]);

    // Prompt user for plaintext
    string plaintext = get_string("plaintext: ");

    // Print the prefix for ciphertext
    printf("ciphertext: ");

    // Iterate over each character in the plaintext
    for (int i = 0, len = strlen(plaintext); i < len; i++)
    {
        // Rotate the character if it's a letter and print it
        printf("%c", rotate(plaintext[i], key));
    }

    // Print a newline after the ciphertext
    printf("\n");

    // Exit successfully
    return 0;
}

// Function to check if a string contains only digits
bool only_digits(string s)
{
    for (int i = 0, len = strlen(s); i < len; i++)
    {
        if (!isdigit(s[i]))
        {
            return false;
        }
    }
    return true;
}

// Function to rotate a character by n positions
char rotate(char c, int n)
{
    // Check if the character is uppercase
    if (isupper(c))
    {
        return 'A' + (c - 'A' + n) % 26;
    }
    // Check if the character is lowercase
    else if (islower(c))
    {
        return 'a' + (c - 'a' + n) % 26;
    }
    // Return the character unchanged if it's not a letter
    return c;
}
```

```c
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

bool only_digits(string text);
char caesar(char c, int key);

int main(int argc, string argv[])
{
    // check the input arguments
    if (argc != 2 || only_digits(argv[1]) == false)
    {
        printf("Usage: ./caesar key\n");
        return 1;
    }
    // initialize the key as an integer and get the plaintext
    int key = atoi(argv[1]);
    string text = get_string("plaintext: ");
    printf("ciphertext: ");
    // print each shifted character
    for (int i = 0, n = strlen(text); i < n; i++)
    {
        printf("%c", caesar(text[i], key));
    }
    printf("\n");
}

bool only_digits(string text)
{
    for (int i = 0, n = strlen(text); i < n; i++)
    {
        if ((int) text[i] < 48 || (int) text[i] > 57)
        {
            return false;
        }
    }
    return true;
}

char caesar(char c, int key)
```

```c
{
    int norm;
    if (c >= 'a' && c <= 'z')
    {
        norm = 97;
    }
    else if (c >= 'A' && c <= 'Z')
    {
        norm = 65;
    }
    else
    {
        return c;
    }
    c -= norm;
    c += key % 26;
    if (c >= 26)
    {
        c -= 26;
    }
    c = norm + c;
    return c;
}
```