Write a complete C program that implements a “circular shift” of an array. Input to the program will be, in order,

- The number of elements $n$ in the array.
- The elements in the array.
- The amount of the circular shift $s$

Output will be the elements in the array after the circular shift.

**Circular Shift**

In a circular shift the elements of the array are shifted right or left with the elements that “fall off” either end being added to the beginning or the end of the array, respectively.

For example, suppose the array $A$ has the following elements

<table>
<thead>
<tr>
<th>Subscript</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

Then a circular shift of 2 would result in the following elements of $A$

<table>
<thead>
<tr>
<th>Subscript</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>10</td>
<td>15</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

So the 1, the 3, and the 6 are shifted to the right 2 places, and the 10 and the 15 are moved, in order, to the beginning of the array. Notice that if $x = A[i]$ in the input array, then

$$x = A[(i+s) \mod n]$$

in the output array.

If we start with the same array $A$

<table>
<thead>
<tr>
<th>Subscript</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

and now do a circular shift of -2, the following array would result:
Now the 6, the 10, and the 15 are shifted to the left 2 places, and the 1 and the 3 are moved, in order, to the end of the array. Notice that when we shift to the left, if \( x = A[i] \) in the input array, then

\[
x = A[(i+s+n) \mod n]
\]

in the output array.

The same ideas work even if the shift amount \( s \) is very large in the positive or negative direction. For example, if \( s = 28 \), we can think of the shift as a sequence of right shifts.

1. Right shift 5
2. Right shift 5
3. Right shift 5
4. Right shift 5
5. Right shift 5
6. Right shift 3

Since there are 5 elements in \( A \), a shift of 5, has no effect on the locations of the elements So we only need to do a shift of 3, and the original array \( A \)

\[
\begin{array}{cccc}
\text{Subscript} & 0 & 1 & 2 & 3 & 4 \\
\text{Element}   & 6 & 10 & 15 & 1 & 3 \\
\end{array}
\]

becomes

\[
\begin{array}{cccc}
\text{Subscript} & 0 & 1 & 2 & 3 & 4 \\
\text{Element}   & 1 & 3 & 6 & 10 & 15 \\
\end{array}
\]

The same reasoning applies with negative shifts, except instead of having right shifts of size \( n \) having no effect, left shifts of size \(-n\) have no effect. (Remember, \( n \) is the number of elements in the array.) So, effectively, the shift that matters is the final shift. If we shift our original array by -28, then we're effectively doing a shift of -3:

\[
\begin{array}{cccc}
\text{Subscript} & 0 & 1 & 2 & 3 & 4 \\
\text{Element}   & 1 & 3 & 6 & 10 & 15 \\
\end{array}
\]

becomes
So the formula we used before to determine where $A[i]$ goes in the new array when $n < 0$ still works, but for the shift we can use $s = \text{original}_s \% n$.

Note that some old C compilers have all remainders $\geq 0$, but for these compilers, the remainder of $-28 \% 5$ will be $+2$ and a circular shift of $+2$ is the same as a circular shift of $-3$. So you can decide whether the shift is to the right or the left by checking whether $s \% n$ is $\geq 0$ or $< 0$, and this will work regardless of whether you have a new or an old compiler.

**Details**

You should declare an array that can store up to 100 ints, and the circular shift should be implemented as a function that takes the array, its size $n$, and the size of the shift $s$, as arguments. The result of the shift should be stored in the original array.

The circular shift function shouldn’t do any input or output (except possibly `DEBUG` output). Input and output should be done in `main`.

You can assume that $1 \leq n \leq 100$, but $s$ can be any integer, positive, negative, or 0.

**Submission**

Create a subdirectory `h3` of your submit directory and copy your source code to this directory:

```
$ scp my_h3.c <my_userid>@stargate.cs.usfca.edu:
$ ssh <my_userid>@stargate.cs.usfca.edu
$ mkdir /home/submit/cs220-01/<my_userid>/h3
$ cp my_h3.c /home/submit/cs220-01/<my_userid>/h3
```