Loops: for loops & while loops

- Using a loop to solve a problem is called *iteration*.

- *while* is similar to *if statements* but *while repeatedly* “loops back” and executes the *statement until the expression is false*.

- General format of a *while* loop:
  ```
  setup statement(s)
  while (expression) {
    body statement(s)
    update statement(s)
  }
  ```

- *for* loops are a “condensed” version of a *while* loop.

Loop: Exercise

Define the following C function: (use iteration)
```
// draw_circle(size) draws a circle inside of a square
// with dimensions (size * 2 + 1)
// requires: size >= 1
// effects: produces output
```

- There is a simple example of the output in "simple.expect"
Integer Overflow: Introduction

- Any variable in C takes up a certain amount of memory (bits).
- This limits the range of values that can be represented.
- Any time you try to go past this limit it is called an “overflow”

Integer Overflow

- A variable of type `int` allocates 32 bits of memory.
- Able to represent negative and positive numbers, so roughly half of this range is negative and roughly half is positive.

As an INT it is impossible to represent outside of the range of:

<table>
<thead>
<tr>
<th>INT_MIN</th>
<th>$-2^{31}$</th>
<th>$-2147482648$</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT_MAX</td>
<td>$2^{31} - 1$</td>
<td>$2147482647$</td>
</tr>
</tbody>
</table>

which is why we have other data-types

Demo: Overflow

Overflow illustrated - Seashell Demo
### Data Types

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Description</th>
<th>Printf</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>Integer (numbers)</td>
<td>%d</td>
</tr>
<tr>
<td>char</td>
<td>Characters</td>
<td>%c</td>
</tr>
<tr>
<td>float</td>
<td>Floating Point decimal numbers</td>
<td>%f</td>
</tr>
<tr>
<td>double</td>
<td>Double precision floating value</td>
<td>%f</td>
</tr>
</tbody>
</table>

### Characters

Characters are integers that are typically used to hold single pieces of text data. They are 8-bit (max value of 127).

```c
char nine = '9';
char not_nine = 9; // Be careful! This is a tab!
char a = 'a';
char also_a = 97; // Equivalent, but bad practice.
char space = ' ';
char newline = '\n'; // Some characters use escape codes.
```

### Character: Exercise

Define the following C function: (use iteration)

```c
swaps(c) changes c from upper to lower case
// (or vice-versa) and leaves non-letter
// values of c unchanged
// requires: size >= 1
// effects: produces output
```