Thoughts after one class
Thoughts on Demon Horde Sort
All your blue sky ideas
→ Notes to wiki at https://robust.cs.unm.edu/
1. **Report in (all, 15 min):** Thoughts after one class. Demon Horde Sort. Your blue sky ideas.

2. **Tell (Dave, all, 15 min):** The arc of the creative process.

3. **Break (all, 5 min):** Turn in signed class form.

4. **Tell and show (Dave, 40 min):** Intro to ulam Part 10: The EventWindow and its contents.

The two attractors of computing

Correct
Efficient

Robust
Correct

“Master of the Universe”

“Member of the Team”
The two attractors of computing

Admissibility requirement
Primary goal

Correct
Efficient

Robust
Correct

Attractor nickname
“Master of the Universe”
“Member of the Team”
The two attractors of computing

- Canonical mechanism
- Admissibility requirement
  - Primary goal

- Algorithm
  - Correct
  - Efficient

- Process
  - Robust
  - Correct

Attractor nickname

- “Master of the Universe”
- “Member of the Team”
Turn in filled-out and signed declaration at the back of the opening day handout!
• In `ulam`, type names must be Capitalized. 'int' is not a type, but 'Int' is.

• There are `primitive` types and `class` types. We'll talk about class types as we get to them.

• Some basic primitive types:
  – `Int` is for signed integers. In many cases the more appropriate choice is `Unsigned`.
  – `Bool` is for a true/false value.
  – There is an additional numeric type called `Unary`.

• Decimal constants are considered signed, but you can put a 'u' at the end to make it unsigned:
  – `12` is a signed constant
  – `12u` is an unsigned constant
EventWindow-related types

- **SiteNum** is a site number in standard order, normally in the range 0u..40u, as in the event window diagram.

- **C2D** is a class used to represent 2D \((x,y)\) coordinates. Each dimension is in the range -32,768..32,767.
  - There are a variety of C2D methods available. We'll talk about them later.

- (Advanced: **Symmetry** is a number in 0u..7u representing the current symmetry in use in the event window.)
Using the EventWindow

- Step 1: Declare an EventWindow instance as a local variable. By convention the instance is called 'ew'
- Step 2: Use 'array notation' and site numbers to read/write to/from the EventWindow.
- Step 3: In addition to array indexing, you can perform other functions by calling EventWindow methods on the instance.
Some EventWindow methods (1)

- **Bool isLegal(SiteNum sn)**
  - Return `true` iff `sn` is a legal event window site (i.e., 0..40)

- **Bool isLive(SiteNum sn)**
  - Return `true` iff `sn` is a usable site in the current event window (e.g., not off the 'edge of the universe').

- **Bool isEmpty(SiteNum sn)**
  - Return `true` iff `sn` is a live site that is currently empty

- **SiteNum size()**
  - Return the number of sites in the event window (41)
Some EventWindow methods (2)

- **Bool ew.swap(SiteNum s1, SiteNum s2)**
  - Swap the contents of `ew[s1]` and `ew[s2]` (if both sites are live).
  - Returns `true` if the swap occurred; `false` if one or both sites weren't live.

- **C2D getCoord(SiteNum sn)**
  - Return the 2D coordinate corresponding to site `sn`.

- **SiteNum getSiteNumber(C2D coord)**
  - Return the site number corresponding to `coord`, if it exists, else return `EventWindow.size()` (which is an illegal SiteNum and will causes event window array accesses to fail if used.)
In **ulam** (like in C/C++), a typedef statement can be used to give an 'alias' to a type.

- The basic format of a typedef statement is
  
  ```
  typedef ExistingType NewTypeName;
  ```

**Example 1:**

```
typedef Int Foo; // Foo is an alias for Int
Foo x;          // ..so x is of type Int
```

**Example 2:**

```
typedef Bool B[3]; // B is an array of 3 Bool
B flags;         // So flags is an array of 3 Bool
if (flags[1]) { ... } // so this is legal
```
Action Items for Monday AUG 28!

1. Join the ulam-users mailing list at
   http://mail.cs.unm.edu/cgi-bin/mailman/listinfo/ulam-users

2. Create a page for yourself at robust.cs.unm.edu. (You log in to the wiki with your CS id/pwd.)

3. Work on Ulam Exercises 10. Email me an .mfz file with your work before 10am next Monday.
Some of these are easier and some are harder; do as many as you can, but don't let yourself get stuck too long!

1. **Check.ulam**: Starting from a single Check atom anywhere in an otherwise empty grid, create a space-filling checkerboard of Check atoms alternating with empty locations.

2. **West.ulam**: Make a West element that moves west until it reaches the edge of the universe, then stops there. Use element metadata \`\symbol {W}` to give if the symbol 'W'.

3. **SwapE.ulam**: Make a blue element SwapE that considers a random immediate neighbor (N,S,E,W) and swaps places with it, but only if the neighbor is empty. (Hint: Use \`\texttt{symmetries all}\` in the element metadata, and also \`\texttt{color \#00f}\` to give it the color blue). What happens if you start with a solid block of SwapE's?

4. **SwapX.ulam**: Make a red element SwapX that acts like SwapE but only swaps if the neighbor is NOT empty. What happens if SwapX's and SwapE's meet?

5. **Cool.ulam**: Make an element Cool that does something that you decide! Be prepared to show us what it does!