Welcome!

Undistinguished Lecture Series

Devin Lange + GradSAC
“What the heck is the Undistinguished Lectures Series?”

- You… probably
Whatever you want it to be!
See Past Projects

https://wiki.cs.utah.edu/undistinguished-lecture-series
This could be you!

Post in #gradsac on slack, or Email gradsac@gmail.com
Tribal Wars, Robots, and Sweet Sweet Productivity Hacks
Outline

• Motivation
• Tribal Wars
• Robots
• Sweet Sweet Productivity Hacks
• Q/A
• Prizes
Outline

- Motivation
- Tribal Wars
- Robots
- Sweet Sweet Productivity Hacks
- Q/A
- Prizes
Covid sucks
Covid sucks

Devin was sad
Outline

• Motivation
• Tribal Wars
• Robots
• Sweet Sweet Productivity Hacks
• Q/A
• Prizes
Tribal Wars / Age

18y
Tribal Wars / Age

18y
Build/Upgrade

Headquarters (Level 6)

In the Headquarters you can construct new buildings or upgrade existing ones. The higher the level of your Headquarters, the faster the constructions will be finished. As soon as your Headquarters is upgraded to level 15, you will be able to demolish buildings in this village (requires 100% loyalty).

<table>
<thead>
<tr>
<th>Construction</th>
<th>Duration</th>
<th>Speed up</th>
<th>Completion</th>
<th>Cancellation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall Level 2</td>
<td>0:01:39</td>
<td>✅ Finish</td>
<td>today 18:15:12</td>
<td>Cancel</td>
</tr>
<tr>
<td>Iron mine Level 4</td>
<td>0:03:22</td>
<td></td>
<td>today 18:18:34</td>
<td>Cancel</td>
</tr>
</tbody>
</table>

Upgrade to a Premium Account to have more than 2 buildings in the queue. » Tell me more...

Buildings

<table>
<thead>
<tr>
<th>Buildings</th>
<th>Requirements</th>
<th>Construct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headquarters Level 6</td>
<td>360 344</td>
<td>Resources available today at 21:10</td>
</tr>
<tr>
<td>Barracks Level 4</td>
<td>504 456</td>
<td>Resources available today at 22:22</td>
</tr>
<tr>
<td>Smithy Level 1</td>
<td>277 230</td>
<td>Resources available today at 20:29</td>
</tr>
<tr>
<td>Rally point Level 1</td>
<td></td>
<td>Building fully constructed</td>
</tr>
<tr>
<td>Statue Level 1</td>
<td></td>
<td>Building fully constructed</td>
</tr>
<tr>
<td>Market Level 1</td>
<td>126 128</td>
<td>Resources available today at 19:14</td>
</tr>
<tr>
<td>Timber camp Level 5</td>
<td>153 202</td>
<td>Resources available today at 19:27</td>
</tr>
<tr>
<td>Clay pit Level 5</td>
<td>215 162</td>
<td>Resources available today at 19:38</td>
</tr>
<tr>
<td>Iron mine Level 3</td>
<td>184 172</td>
<td>Resources available today at 19:53</td>
</tr>
<tr>
<td>Farm Level 2</td>
<td>76 70</td>
<td>Resources available today at 18:49</td>
</tr>
<tr>
<td>Warehouse Level 4</td>
<td>154 130</td>
<td>Resources available today at 19:28</td>
</tr>
<tr>
<td>Hiding place Level 3</td>
<td>98 117</td>
<td>Resources available today at 19:00</td>
</tr>
<tr>
<td>Wall Level 1</td>
<td>79 163</td>
<td>Resources available today at 18:51</td>
</tr>
</tbody>
</table>

Not yet available
Recruit Troops

### Barracks (Level 4)

In the barracks you can recruit infantry. The higher its level the faster the recruitment of troops will be finished.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Requirements</th>
<th>In the village/total</th>
<th>Recruit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spear fighter</td>
<td><img src="50" alt="50" /> <img src="30" alt="30" /> <img src="10" alt="10" /> <img src="1" alt="1" /> <img src="" alt="0:04:30" /></td>
<td>32/32</td>
<td>Resources available today at 16:36</td>
</tr>
<tr>
<td>Swordsman</td>
<td><img src="30" alt="30" /> <img src="30" alt="30" /> <img src="70" alt="70" /> <img src="1" alt="1" /> <img src="" alt="0:06:37" /></td>
<td>20/20</td>
<td>Resources available today at 18:26</td>
</tr>
</tbody>
</table>

Not yet available:
- Axeman
- Archer

- Smithy (Level 2)
- Barracks (Level 5)
- Smithy (Level 5)
Attack!!!!
Attack!!!!
Attack!!!!
Attack!!!!
Player Villages
Empty Villages
Scavenging
Outline

- Motivation
- Tribal Wars
- **Robots**
- Sweet Sweet Productivity Hacks
- Q/A
- Prizes
Long story short, I made a Python script to click those buttons.
```python
running = True
while running:
    try:
        executeNextCommand()
    except:
        log('Failed to execute command')
        cleanupAfterException()
        return
    log('next command = ' + ', '.join(peakCommand()))
    try:
        minToCommand = getMinutesUntilNextCommandAvailable()
    except:
        log('Failed to get time until next command')
        cleanupAfterException()
        return
    log('time to next command = ' + 'sec' + minToCommand)
    if minToCommand == 0:
        continue
    try:
        minToScavenge = getMinutesUntilCanScavenge()
    except:
        log('Failed to get time to scavenge')
        cleanupAfterException()
        return
    minUntilNextCommand = min(minToCommand, minToScavenge)
    log('time to scavenge = ' + 'sec' + minUntilNextCommand)
    if minToScavenge == minUntilNextCommand:
        pushCommand('scavenge')
    if minUntilNextCommand > 0:
        scheduleNext(minUntilNextCommand)
        running = False
    try:
        logResourcesAndPopulation()
    except:
        log('Failed to get resources and population')
    driver.quit()
    return
```

Sample output:
```
2020-06-25 20:46:08.521735
----- PROCESS LAUNCHED -----
first load
logged in
world selected
scavenge
scavenger split for 1235 spears = [464, 408, 363]
464 scavengers sent to level 1
next command = recruit, spear, 23
time to next command = 20
scavenger split for 1235 spears = [464, 408, 363]
time to scavenge = 96
next run in: now + 20 minutes
```
```
def executeNextCommand():
    try:
        minToCommand = getMinutesUntilNextCommandAvailable()
    except:
        log('failed to get time until next command')
        cleanupAfterException()
        return

    log('time to next command = ' + str(minToCommand))
    if minToCommand == 0:
        continue

    minToScavenge = getTimeUntilCanScavenge()
    except:
        log('failed to get time to scavenge')
        cleanupAfterException()
        return

    minUntilNextCommand = min(minToCommand, minToScavenge)
    log('time to scavenge = ' + str(minToScavenge))
    if minToScavenge <= minToCommand:
        pushCommand('scavenge')
    elif minUntilNextCommand > 0:
        scheduleNext(minUntilNextCommand)
        running = False

    try:
        logResourcesAndPopulation()
    except:
        log('failed to get resources and population')
    driver.quit()
    return
def build(buildingType: str) -> bool:
    try:
        readyToBuild = canBuild(buildingType)
    except:
        log('Building likely fully constructed. Skipping')
    return True

    if not readyToBuild:
        log('Can’t build – rescheduling')
        return False

    buildTypeEnum = BuildingTypes[buildingType]
    driver.get(HEADQUARTERS_URL.format(VILLAGE_ID))
    buildingName = INTERNAL_BUILDING_NAMES[buildTypeEnum]
    element = wait_until(EC.element_to_be_clickable((By.XPATH, element.click))
    log('built ' + buildingType)
    return True
running = True
while running:
    try:
        executeNextCommand()
    except:
        log('Failed to execute command')
        cleanupAfterException()
        log('next command = ' + ', '.join(peekCommand()))
        return
    try:
        minToCommand = getMinutesUntilNextCommandAvailable()
    except:
        log('failed to get time until next command')
        cleanupAfterException()
        log('time to next command = ' + str(minToCommand))
        if minToCommand == 0:
            continue
    try:
        minToScavenge = getTimeUntilCanScavenge()
    except:
        log('failed to get time to scavenge')
        cleanupAfterException()
        log('time to scavenge = ' + str(minToScavenge))
    minUntilNextCommand = min(minToCommand, minToScavenge)
    log('time to scavenge = ' + str(minToScavenge))
    if minToScavenge <= minToCommand:
        pushCommand('scavenge')
    if minUntilNextCommand > 0:
        scheduleNext(minUntilNextCommand)
        running = False
    try:
        logResourcesAndPopulation()
    except:
        log('failed to get resources and population')
        driver.quit()
    return
debug.executeNextCommand()
running = True
while running:
    try:
        executeNextCommand()
    except:
        log(’Failed to execute command’)
    cleanupAfterException()
    log(’next command = ’ + ’,’.join(peakCommand))
    return
    try:
        minToCommand = getMinutesUntilNextCommandAvailable()
    except:
        log(’failed to get time until next command’)
    cleanupAfterException()
    log(’time to next command = ’ + str(minToCommand))
    if minToCommand == 0:
        continue
    try:
        minToScavenge = getTimeUntilCanScavenge()
    except:
        log(’failed to get time to scavenge’)
    cleanupAfterException()
    minUntilNextCommand = min(minToCommand, minToScavenge)
    log(’time to scavenge = ’ + str(minToScavenge))
    if minToScavenge == minToCommand:
        pushCommand(’scavenge’)
running = True
while running:
    try:
        executeNextCommand()
    except:
        log('Failed to execute command')
        cleanupAfterException()
    log('next command = ' + ','.join(peekCommand()))
    return
    try:
        minToCommand = getMinutesUntilNextCommandAvailable()
    except:
        log('Failed to get time until next command')
        cleanupAfterException()
    log('time to next command = ' + str(minToCommand))
    if minToCommand == 0:
        continue
    try:
        minToScavenge = getTimeUntilCanScavenge()
    except:
        log('Failed to get time to scavenge')
        cleanupAfterException()
    log('time to scavenge = ' + str(minToScavenge))
    minUntilNextCommand = min(minToCommand, minToScavenge)
    log('time to scavenge = ' + str(minUntilNextCommand))
    if minUntilNextCommand == 0:
        pushCommand('scavenge')
if minUntilNextCommand > 0:
    scheduleNext(minUntilNextCommand)
    running = False

try:
    logResourcesAndPopulation()
except:
    log('Failed to get resources and population')
driver.quit()
def scheduleNext(minutesInFuture):
    if minutesInFuture == 0:
        nextRunTime = 'now'
    else:
        nextRunTime = 'now + ' + str(minutesInFuture) + ' minutes'
    log('next run in: ' + nextRunTime)
    nexttime = str(datetime.datetime.now() + datetime.timedelta(minutes=minutesInFuture))
    log(nextTime, True, True)
    os.system('echo ~/Projects/minis/tribalWars/runCommand.sh | sudo at ' + nextRunTime)
    return

if minutesInNextCommand > 0:
    scheduleNext(minutesInNextCommand)
running = False
def scheduleNext(minutesInFuture):
    if minutesInFuture == 0:
        nextRunTime = 'now'
    else:
        nextRunTime = 'now + ' + str(minutesInFuture) + ' minutes'
    log('next run in: ' + nextRunTime)
    nexttime = str(datetime.datetime.now() + datetime.timedelta(minutes=minutesInFuture))
    log(nexttime, True, True)
    os.system('echo ~/Projects/minis/tribalWars/runCommand.sh | sudo at ' + nextRunTime)
    return

if minToCommand == 0:
    continue

try:
    minToScavenge = getTimeUntilCanScavenge()
except:
    log('failed to get time to scavenge')
    cleanupAfterException()
    return

minUntilNextCommand = min(minToCommand, minToScavenge)
log('time to scavenge = ' + str(minToScavenge))

if minToScavenge == minToCommand:
    pushCommand('scavenge')

if minUntilNextCommand > 0:
    scheduleNext(minUntilNextCommand)
    running = False
'at'
command
security
Rally point (Level 1)

On the rally point your fighters meet. Here you can command your armies.

Scavenging

Send your troops away to scavenge for resources surrounding your village. The units will return when they collect enough resources - but have in mind once you send the troops away you can’t recall them.

Lockeysical Looters
Collectible Resources

Humble Haulers
Collectible Resources

Clever Collectors

Great Gatherers

Start -20%
Unlock

Start -20%
Unlock

Server time: 19:05 17/10/2021
LootFactor = 0.1
LootFactor = 0.25
LootFactor = 0.5
LootFactor = 0.75

Rally point (Level 1)
On the rally point your fighters meet. Here you can command your armies.

Scavenging
Send your troops away to scavenge for resources surrounding your village. The units will return when they collect enough resources - but have in mind once you send the troops away you can’t recall them.

ResourcePerSecond = \frac{\text{CarryCapacity} \cdot \text{LootFactor}}{\text{TimeToScavange}}
ResourcePerSecond = \frac{\text{CarryCapacity} \cdot \text{LootFactor}}{\text{TimeToScavange}}

\text{TimeToScavange} = \text{SpeedFactor} \cdot ((\text{CarryCapacity}^2 \cdot 100 \cdot \text{LootFactor}^2)^{\text{exp}} + \text{InitSeconds})
ResourcePerSecond = \frac{CarryCapacity \cdot LootFactor}{TimeToScavenge}

TimeToScavenge = SpeedFactor \cdot ((CarryCapacity^2 \cdot 100 \cdot LootFactor^2)^{0.45} + InitSeconds)

0.9045869428 1800
ResourcePerSecond = \frac{CarryCapacity \cdot LootFactor}{TimeToScavenge}

TimeToScavenge = SpeedFactor \cdot ((CarryCapacity^2 \cdot 100 \cdot LootFactor^2)^{0.45} + InitSeconds)^{0.9045869428} + 1800
```python
def getOptimalScavengerSplit(): -> List[int]:
    totalSpearCount = getTroopCount('spear')
    totalSpearCount = min(totalSpearCount, 2000)  # only budget so many spears for scavenging
    numUnlocked = getNumberOfAvailableScavengeLevels()
    weightList = [1 / float(numUnlocked) for _ in range(numUnlocked)]
    weightList = [round(w * totalSpearCount) for w in weightList]
    weightList = [totalSpearCount - sum(weightList[i:]) for i in range(len(weightList))]
    resPerHour = calculateResourcesPerHour(totalSpearCount, weightList)
    i = 0
    stepSize = 1 / float(resPerHour)
    lastLogRate = resPerHour
    maxCount = 10000
    for _ in range(maxCount):
        i += 1
        weightListOption1 = [w for w in weightList]
        weightListOption2 = [w for w in weightList]
        weightListOption3 = [w for w in weightList]
        thisStepSize1 = min(stepSize, weightList[i] - weightList[i])
        weightListOption1[i] = thisStepSize1
        weightListOption2[i] = thisStepSize1
        weightListOption3[i] = thisStepSize1
        resPerHour1 = calculateResourcesPerHour(totalSpearCount, weightListOption1)
        resPerHour2 = calculateResourcesPerHour(totalSpearCount, weightListOption2)
        resPerHour3 = calculateResourcesPerHour(totalSpearCount, weightListOption3)
        if resPerHour1 > resPerHour2 and resPerHour1 > resPerHour3:
            weightList = weightListOption1
            resPerHour = resPerHour1
        elif resPerHour2 > resPerHour3:
            weightList = weightListOption2
            resPerHour = resPerHour2
        i = 12
        if i == 0:
            if resPerHour == lastLogRate:
                break
            else:
                lastLogRate = resPerHour
        troopCount = [round(w * totalSpearCount) for w in weightList]
        toDistribute = 0
        toIndex = 0
        for i, count in enumerate(troopCount):
            if count < 10:
                toDistribute += count
                troopCount[i] = 0
            else:
                toIndex += 1
                if toDistribute > 0:
                    troopCount[toIndex] += toDistribute
                    log('scavenger split for {} spears = {}'.format(totalSpearCount, troopCount))
        return troopCount
```
<table>
<thead>
<tr>
<th></th>
<th>All troops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lackadaisical Looters</td>
<td>(0)</td>
</tr>
<tr>
<td>Humble Haulers</td>
<td>(0)</td>
</tr>
<tr>
<td>Clever Collectors</td>
<td>(0)</td>
</tr>
<tr>
<td>Great Gatherers</td>
<td>(1)</td>
</tr>
</tbody>
</table>

24 → 26

25

25
Limitations

- Code currently only supports scavenging with spear troop type.
Limitations

- Code currently only supports scavenging with spear troop type.
Limitations

- Code currently only supports scavenging with spear troop type.

- While you are growing your total spear count you will have a few spears not scavenging.
Limitations

- Code currently only supports scavenging with *spear* troop type.
- While you are growing your total spear count you will have a few spears not scavenging.
Limitations

- Code currently only supports scavenging with **spear** troop type.
- While you are growing your total spear count you will have a few spears not scavenging.
Limitations

- Code currently only supports scavenging with spear troop type.
- While you are growing your total spear count, you will have a few spears not scavenging.

There are no limitations
Motivation
Covid sucks

Devin was sad
Covid sucks

Devin was sad
Covid sucks

Devin was sad
Covid sucks

Devin was sad distracted
Outline

• Motivation
• Tribal Wars
• Robots
• **Sweet Sweet Productivity Hacks**
• Q/A
• Prizes
“What if I was a robot? How would I write code to optimize myself?”

- Devin
Hypothetical Execution
Hypothetical Execution

Computer programs are typically written so that computers can run them and execute some logic. Hypothetical execution is the idea that programming languages are useful for encoding some sequence of actions and logic, but will never actually be run by a computer.
Event Hooks

```python
enum OncePerDayEvents
{
    StartUp = 'StartUp',
    ShutDown = 'ShutDown'
}

enum OncePerWorkdayEvents
{
    StartWork = 'StartWork',
    StartLunchBreak = 'StartLunchBreak',
    EndLunchBreak = 'EndLunchBreak',
    EndWork = 'EndWork'
}

enum FrequentEvents
{
    EnterKitchen = 'EnterKitchen',
    ChoreBreak = 'ChoreBreak'
}
```
enum OncePerDayEvents
{
    StartUp = 'StartUp',
    ShutDown = 'ShutDown'
}

enum OncePerWorkdayEvents
{
    StartWork = 'StartWork',
    StartLunchBreak = 'StartLunchBreak',
    EndLunchBreak = 'EndLunchBreak',
    EndWork = 'EndWork'
}

enum FrequentEvents
{
    EnterKitchen = 'EnterKitchen',
    ChoreBreak = 'ChoreBreak'
}

export class MyRoutines
{
    private _worldState : WorldState;
    public get worldState(): WorldState {
        return this._worldState;
    }

    private _dailyChoreList : Function[];
    public get dailyChoreList(): Function[] {
        return this._dailyChoreList;
    }

    public constructor()
    {
        // Daily events
        document.addEventListener(OncePerDayEvents.StartUp, this.OnStartUp);
        document.addEventListener(OncePerDayEvents.ShutDown, this.OnShutDown);

        // Daily Workday events
        document.addEventListener(OncePerWorkdayEvents.StartWork, this.OnStartWork);
        document.addEventListener(OncePerWorkdayEvents.StartLunchBreak, this.OnStartLunchBreak);
        document.addEventListener(OncePerWorkdayEvents.EndLunchBreak, this.OnEndLunchBreak);
        document.addEventListener(OncePerWorkdayEvents.EndWork, this.OnEndWork);

        // FrequentEvents
        document.addEventListener(FrequentEvents.EnterKitchen, this.OnEnterKitchen);
        document.addEventListener(FrequentEvents.ChoreBreak, this.doChore);
    }
}
Event Handling

class MyRoommates
{
    private _worldState : WorldState;
    public get worldState(): WorldState {
        return this._worldState;
    }
}

private _dailyChoreList : Function[];
public get dailyChoreList(): Function[] {
    return this._dailyChoreList;
}

public constructor()
{
    // Daily events
    document.addEventListener(OncePerDayEvents.StartUp, this.onStartUp);
    document.addEventListener(OncePerDayEvents.ShutDown, this.onShutDown);
    document.addEventListener(OncePerDayEvents.StartWork, this.onStartWork);
    document.addEventListener(OncePerDayEvents.EndWork, this.onEndWork);
    document.addEventListener(OncePerDayEvents.StartLunchBreak, this.onStartLunchBreak);
    document.addEventListener(OncePerDayEvents.EndLunchBreak, this.onEndLunchBreak);
    document.addEventListener(OncePerDayEvents.Shutdown, this.onShutdown);
    document.addEventListener(OncePerDayEvents.EnterKitchen, this.onEnterKitchen);
    document.addEventListener(OncePerDayEvents.ChoreBreak, this.doChore);

    // Weekly events
    document.addEventListener(OncePerWeekEvents.Saturday, this.onSaturday);
    document.addEventListener(OncePerWeekEvents.Sunday, this.onSunday);
    document.addEventListener(OncePerWeekEvents.WorkDay, this.onWorkDay);
    document.addEventListener(OncePerWeekEvents.WorkNight, this.onWorkNight);
    document.addEventListener(OncePerWeekEvents.WorkDayEnd, this.onWorkDayEnd);
    document.addEventListener(OncePerWeekEvents.WorkNightEnd, this.onWorkNightEnd);

    public OnStartUp(): void
    {
        this.initDailyChoreList();
        this.openEyes();
        this.praiseTheSun();
        this.applyDeodorant();
        if (this.worldState.currentSeason === Seasons.Summer)
        {
            this.turnACtoPowerSave();
        }
        this.getDressed();
        this.makeBed();
        this.startUpFirst();
        this.brushTeeth();
        const outOfEggs = this.worldState.cookedEggCount === 0
        if (valueOfEggs)
        {
            this.startEggCooking();
        }
        this.initCoffee();
        if (!valueOfEggs)
        {
            if (this.worldState.cleanDishesPresent && this.worldState.cleanDishesDry)
            {
                this.putCleanDishesAway();
            }
        }
        else
        {
            this.prepareEggMix();
            if (this.worldState.dirtyDishesPresent)
            {
                this.washLoadOfDishes();
            }
        }
    this.eatBreakfast();
}
Event Handling

```java
public OnStartup(): void {
    this.initDailyChoreList();
    this.openEyes();
    this.praiseTheSun();
    this.applyDeodorant();
    if (this.worldState.currentSeason == Seasons.Summer) {
        this.turnACtoPowerSave();
    }
    this.getDressed();
    this.makeBed();
    this.startUpFirst();
    this.brushTeeth();
    const outOfEggs = this.worldState.cookedEggCount == 0
    if (OutOfEggs) {
        this.startEggCooking();
    }
    this.initCoffee();
    if (!OutOfEggs) {
        if (this.worldState.cleanDishesPresent && this.worldState.cleanDishesDry) {
            this.putCleanDishesAway();
        } else {
            this.prepareEggLute();
            if (this.worldState.dirtyDishesPresent) {
                this.washLoadOfDishes();
            }
        }
    }
    this.eatBreakfast();
}
```
Finding the right level of abstraction

```java
private void initDailyChoreList()
{
    this._dailyChoreList =
    [this.loadWashingMachine,
    this.scoopCatLitter,
    this.cleanOutRoomba,
    this.takeOutGarbage,
    this.washLoadOfDishes,
    this.handleNonRecurringTasks,
    this.handleRecurringTasks,
    this.reviewPersonalProjects,
    this.handleRecurringTasks
    ];
}

public void doChore()
{
    if (this.dailyChoreList.length == 0)
    {
        return;
    }

    let nextChore = this.dailyChoreList.shift();
    nextChore();
}
Finding the right level of abstraction

```java
private initDailyChoreList(): void {
    this._dailyChoreList = [
        this.loadWashingMachine,
        this.scoopCatLitter,
        this.cleanRoomba,
        this.takeOutGarbage,
        this.washLoadOfDishes,
        this.handleNonRecurringTasks,
        this.handleRecurringTasks,
        this.reviewPersonalProjects,
        this.handleRecurringTasks
    ];
}

public doChore(): void {
    if (this.dailyChoreList.length == 0) {
        return;
    }
    let nextChore = this.dailyChoreList.shift();
    nextChore();
}
```
<table>
<thead>
<tr>
<th>Task</th>
<th>Last Done</th>
<th>Due Date</th>
<th>Cadence</th>
<th>Owner</th>
<th>Type</th>
<th>Calendar ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean surface of Washer/Dryer</td>
<td>12/6/2020</td>
<td>12/20/2020</td>
<td>14</td>
<td>Devin</td>
<td>House</td>
<td>49t2q4th6lane</td>
</tr>
<tr>
<td>Wash Bed Sheets</td>
<td>12/5/2020</td>
<td>1/9/2021</td>
<td>35</td>
<td>Devin</td>
<td>House</td>
<td>kj093hojamtcf</td>
</tr>
<tr>
<td>Cleanup Devin's office</td>
<td>1/10/2021</td>
<td>1/17/2021</td>
<td>7</td>
<td>Devin</td>
<td>House</td>
<td>107isssses08q</td>
</tr>
<tr>
<td>Clean french press</td>
<td>1/11/2021</td>
<td>1/18/2021</td>
<td>7</td>
<td>Devin</td>
<td>House</td>
<td>1s45u3a64qjs</td>
</tr>
<tr>
<td>Remove basement cobwebs</td>
<td>1/5/2021</td>
<td>1/19/2021</td>
<td>14</td>
<td>Devin</td>
<td>House</td>
<td>74cfsjn5u2n3c</td>
</tr>
<tr>
<td>Wash Pillow Covers</td>
<td>1/5/2021</td>
<td>1/25/2021</td>
<td>20</td>
<td>Devin</td>
<td>House</td>
<td>hpu94obnr9d:</td>
</tr>
<tr>
<td>Vacuum floor under cat litter</td>
<td>1/12/2021</td>
<td>1/26/2021</td>
<td>14</td>
<td>Devin</td>
<td>House</td>
<td>ao6dd27s0ad:</td>
</tr>
<tr>
<td>Fun Hike</td>
<td>1/16/2021</td>
<td>1/26/2021</td>
<td>10</td>
<td>Devin</td>
<td>Personal</td>
<td>n4b6st012tg4t</td>
</tr>
<tr>
<td>Clean garbage disposal</td>
<td>12/5/2020</td>
<td>1/30/2021</td>
<td>56</td>
<td>Devin</td>
<td>House</td>
<td>2v39n2n501aj</td>
</tr>
<tr>
<td>Test Fire alarms</td>
<td>1/11/2021</td>
<td>2/1/2021</td>
<td>21</td>
<td>Devin</td>
<td>House</td>
<td>5mequc998n0c</td>
</tr>
<tr>
<td>Deep clean kitty litter</td>
<td>1/5/2021</td>
<td>2/2/2021</td>
<td>28</td>
<td>Devin</td>
<td>House</td>
<td>6pndk0eccsect</td>
</tr>
<tr>
<td>Deep clean roomba</td>
<td>2/1/2021</td>
<td>2/8/2021</td>
<td>7</td>
<td>Devin</td>
<td>House</td>
<td>7sf3frjrrdef5p:</td>
</tr>
<tr>
<td>Reconnect with friend</td>
<td>2/3/2021</td>
<td>2/10/2021</td>
<td>7</td>
<td>Devin</td>
<td>Personal</td>
<td>2tbo cru6e6b:</td>
</tr>
<tr>
<td>Eat at dinner table</td>
<td>2/3/2021</td>
<td>2/10/2021</td>
<td>7</td>
<td>Devin</td>
<td>Personal</td>
<td>efk4k0va1a7vi:</td>
</tr>
<tr>
<td>Cut fingernails</td>
<td>2/4/2021</td>
<td>2/11/2021</td>
<td>7</td>
<td>Devin</td>
<td>Personal</td>
<td>1bmtjvoq5djv2:</td>
</tr>
<tr>
<td>Clean shower drain</td>
<td>2/2/2021</td>
<td>2/23/2021</td>
<td>21</td>
<td>Devin</td>
<td>House</td>
<td>drk967je4vsn</td>
</tr>
<tr>
<td><strong>Replace Furnace Filter</strong></td>
<td>1/27/2021</td>
<td>2/26/2021</td>
<td>30</td>
<td>Devin</td>
<td>House</td>
<td>3g7eu6h9qjn</td>
</tr>
<tr>
<td>Clean Furnace Area</td>
<td>12/6/2020</td>
<td>2/28/2021</td>
<td>84</td>
<td>Devin</td>
<td>House</td>
<td>2uvppppqnub1:</td>
</tr>
<tr>
<td>Mow Lawn</td>
<td>3/1/2021</td>
<td>3/8/2021</td>
<td>7</td>
<td>Devin</td>
<td>House</td>
<td>oohvby6d4dx</td>
</tr>
<tr>
<td>Clean House Vents</td>
<td>12/10/2020</td>
<td>3/11/2021</td>
<td>91</td>
<td>Devin</td>
<td>House</td>
<td>08afpagmdafc</td>
</tr>
<tr>
<td>Schedule next dentist appt</td>
<td>10/27/2020</td>
<td>9/21/2021</td>
<td>329</td>
<td>Devin</td>
<td>Personal</td>
<td>pkrhpt6lpg6f</td>
</tr>
<tr>
<td>Schedule next eye exam</td>
<td>10/27/2020</td>
<td>9/21/2021</td>
<td>329</td>
<td>Devin</td>
<td>Personal</td>
<td>806751rfllda1l</td>
</tr>
</tbody>
</table>
Outline

- Motivation
- Tribal Wars
- Robots
- Sweet Sweet Productivity Hacks
- Q/A
- Prizes
Any questions?
Outline

• Motivation
• Tribal Wars
• Robots
• Sweet Sweet Productivity Hacks
• Q/A
• Prizes
Prize + Raffle!