export const Moves: {[moveid: string]: MoveData} = {

"10000000voltthunderbolt": {

num: 719,

accuracy: true,

basePower: 195,

category: "Special",

desc: "Has a very high chance for a critical hit.",

shortDesc: "Very high critical hit ratio.",

isNonstandard: "Past",

name: "10,000,000 Volt Thunderbolt",

pp: 1,

priority: 0,

flags: {},

isZ: "pikashuniumz",

critRatio: 3,

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

absorb: {

num: 71,

accuracy: 100,

basePower: 20,

category: "Special",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Absorb",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Grass",

contestType: "Clever",

},

accelerock: {

num: 709,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Accelerock",

pp: 20,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Cool",

},

acid: {

num: 51,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the foe(s) Sp. Def by 1.",

name: "Acid",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "allAdjacentFoes",

type: "Poison",

contestType: "Clever",

},

acidarmor: {

num: 151,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 2 stages.",

shortDesc: "Raises the user's Defense by 2.",

name: "Acid Armor",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 2,

},

secondary: null,

target: "self",

type: "Poison",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

aciddownpour: {

num: 628,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Acid Downpour",

pp: 1,

priority: 0,

flags: {},

isZ: "poisoniumz",

secondary: null,

target: "normal",

type: "Poison",

contestType: "Cool",

},

acidspray: {

num: 491,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 100% chance to lower the target's Special Defense by 2 stages.",

shortDesc: "100% chance to lower the target's Sp. Def by 2.",

name: "Acid Spray",

pp: 20,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spd: -2,

},

},

target: "normal",

type: "Poison",

contestType: "Beautiful",

},

acrobatics: {

num: 512,

accuracy: 100,

basePower: 55,

basePowerCallback(pokemon, target, move) {

if (!pokemon.item) {

this.debug("Power doubled for no item");

return move.basePower \* 2;

}

return move.basePower;

},

category: "Physical",

shortDesc: "Power doubles if the user has no held item.",

name: "Acrobatics",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

acupressure: {

num: 367,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises a random stat by 2 stages as long as the stat is not already at stage 6. The user can choose to use this move on itself or an adjacent ally. Fails if no stat stage can be raised or if used on an ally with a substitute.",

shortDesc: "Raises a random stat of the user or an ally by 2.",

name: "Acupressure",

pp: 30,

priority: 0,

flags: {},

onHit(target) {

const stats: BoostName[] = [];

let stat: BoostName;

for (stat in target.boosts) {

if (target.boosts[stat] < 6) {

stats.push(stat);

}

}

if (stats.length) {

const randomStat = this.sample(stats);

const boost: SparseBoostsTable = {};

boost[randomStat] = 2;

this.boost(boost);

} else {

return false;

}

},

secondary: null,

target: "adjacentAllyOrSelf",

type: "Normal",

zMove: {effect: 'crit2'},

contestType: "Tough",

},

aerialace: {

num: 332,

accuracy: true,

basePower: 60,

category: "Physical",

shortDesc: "This move does not check accuracy.",

name: "Aerial Ace",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

aeroblast: {

num: 177,

accuracy: 95,

basePower: 100,

category: "Special",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

isNonstandard: "Past",

name: "Aeroblast",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, distance: 1},

critRatio: 2,

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

afteryou: {

num: 495,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target makes its move immediately after the user this turn, no matter the priority of its selected move. Fails if the target would have moved next anyway, or if the target already moved this turn.",

shortDesc: "The target makes its move right after the user.",

name: "After You",

pp: 15,

priority: 0,

flags: {authentic: 1, mystery: 1},

onHit(target) {

if (target.side.active.length < 2) return false; // fails in singles

const action = this.queue.willMove(target);

if (action) {

this.queue.prioritizeAction(action);

this.add('-activate', target, 'move: After You');

} else {

return false;

}

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Cute",

},

agility: {

num: 97,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Speed by 2 stages.",

shortDesc: "Raises the user's Speed by 2.",

name: "Agility",

pp: 30,

priority: 0,

flags: {snatch: 1},

boosts: {

spe: 2,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cool",

},

aircutter: {

num: 314,

accuracy: 95,

basePower: 60,

category: "Special",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio. Hits adjacent foes.",

name: "Air Cutter",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "allAdjacentFoes",

type: "Flying",

contestType: "Cool",

},

airslash: {

num: 403,

accuracy: 95,

basePower: 75,

category: "Special",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Air Slash",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, distance: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "any",

type: "Flying",

contestType: "Cool",

},

alloutpummeling: {

num: 624,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "All-Out Pummeling",

pp: 1,

priority: 0,

flags: {},

isZ: "fightiniumz",

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

allyswitch: {

num: 502,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps positions with its ally. Fails if the user is the only Pokemon on its side.",

shortDesc: "The user swaps positions with its ally.",

name: "Ally Switch",

pp: 15,

priority: 2,

flags: {},

onTryHit(source) {

if (source.side.active.length === 1) return false;

if (source.side.active.length === 3 && source.position === 1) return false;

},

onHit(pokemon) {

const newPosition = (pokemon.position === 0 ? pokemon.side.active.length - 1 : 0);

if (!pokemon.side.active[newPosition]) return false;

if (pokemon.side.active[newPosition].fainted) return false;

this.swapPosition(pokemon, newPosition, '[from] move: Ally Switch');

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

amnesia: {

num: 133,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Defense by 2 stages.",

shortDesc: "Raises the user's Sp. Def by 2.",

name: "Amnesia",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

spd: 2,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

anchorshot: {

num: 677,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Prevents the target from switching out.",

name: "Anchor Shot",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

onHit(target, source, move) {

if (source.isActive) target.addVolatile('trapped', source, move, 'trapper');

},

},

target: "normal",

type: "Steel",

contestType: "Tough",

},

ancientpower: {

num: 246,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 10% chance to raise the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage.",

shortDesc: "10% chance to raise all stats by 1 (not acc/eva).",

name: "Ancient Power",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

self: {

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

},

},

target: "normal",

type: "Rock",

contestType: "Tough",

},

appleacid: {

num: 787,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 100% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "100% chance to lower the target's Sp. Def by 1.",

name: "Apple Acid",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Grass",

},

aquajet: {

num: 453,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Aqua Jet",

pp: 20,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

contestType: "Cool",

},

aquaring: {

num: 392,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user has 1/16 of its maximum HP, rounded down, restored at the end of each turn while it remains active. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down. If the user uses Baton Pass, the replacement will receive the healing effect.",

shortDesc: "User recovers 1/16 max HP per turn.",

name: "Aqua Ring",

pp: 20,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'aquaring',

condition: {

onStart(pokemon) {

this.add('-start', pokemon, 'Aqua Ring');

},

onResidualOrder: 6,

onResidual(pokemon) {

this.heal(pokemon.baseMaxhp / 16);

},

},

secondary: null,

target: "self",

type: "Water",

zMove: {boost: {def: 1}},

contestType: "Beautiful",

},

aquatail: {

num: 401,

accuracy: 90,

basePower: 90,

category: "Physical",

shortDesc: "No additional effect.",

name: "Aqua Tail",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

armthrust: {

num: 292,

accuracy: 100,

basePower: 15,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Arm Thrust",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

aromatherapy: {

num: 312,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Every Pokemon in the user's party is cured of its major status condition. Active Pokemon with the Sap Sipper Ability are not cured, unless they are the user.",

shortDesc: "Cures the user's party of all status conditions.",

name: "Aromatherapy",

pp: 5,

priority: 0,

flags: {snatch: 1, distance: 1},

onHit(pokemon, source, move) {

this.add('-activate', source, 'move: Aromatherapy');

let success = false;

for (const ally of pokemon.side.pokemon) {

if (ally !== source && ((ally.hasAbility('sapsipper')) ||

(ally.volatiles['substitute'] && !move.infiltrates))) {

continue;

}

if (ally.cureStatus()) success = true;

}

return success;

},

target: "allyTeam",

type: "Grass",

zMove: {effect: 'heal'},

contestType: "Clever",

},

aromaticmist: {

num: 597,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the target's Special Defense by 1 stage. Fails if there is no ally adjacent to the user.",

shortDesc: "Raises an ally's Sp. Def by 1.",

name: "Aromatic Mist",

pp: 20,

priority: 0,

flags: {authentic: 1},

boosts: {

spd: 1,

},

secondary: null,

target: "adjacentAlly",

type: "Fairy",

zMove: {boost: {spd: 2}},

contestType: "Beautiful",

},

assist: {

num: 274,

accuracy: true,

basePower: 0,

category: "Status",

desc: "A random move among those known by the user's party members is selected for use. Does not select Assist, Baneful Bunker, Beak Blast, Belch, Bestow, Bounce, Celebrate, Chatter, Circle Throw, Copycat, Counter, Covet, Destiny Bond, Detect, Dig, Dive, Dragon Tail, Endure, Feint, Fly, Focus Punch, Follow Me, Helping Hand, Hold Hands, King's Shield, Mat Block, Me First, Metronome, Mimic, Mirror Coat, Mirror Move, Nature Power, Phantom Force, Protect, Rage Powder, Roar, Shadow Force, Shell Trap, Sketch, Sky Drop, Sleep Talk, Snatch, Spiky Shield, Spotlight, Struggle, Switcheroo, Thief, Transform, Trick, Whirlwind, or any Z-Move.",

shortDesc: "Uses a random move known by a team member.",

isNonstandard: "Past",

name: "Assist",

pp: 20,

priority: 0,

flags: {},

onHit(target) {

const noAssist = [

'assist', 'banefulbunker', 'beakblast', 'belch', 'bestow', 'bounce', 'celebrate', 'chatter', 'circlethrow', 'copycat', 'counter', 'covet', 'destinybond', 'detect', 'dig', 'dive', 'dragontail', 'endure', 'feint', 'fly', 'focuspunch', 'followme', 'helpinghand', 'holdhands', 'kingsshield', 'matblock', 'mefirst', 'metronome', 'mimic', 'mirrorcoat', 'mirrormove', 'naturepower', 'phantomforce', 'protect', 'ragepowder', 'roar', 'shadowforce', 'shelltrap', 'sketch', 'skydrop', 'sleeptalk', 'snatch', 'spikyshield', 'spotlight', 'struggle', 'switcheroo', 'thief', 'transform', 'trick', 'whirlwind',

];

const moves = [];

for (const pokemon of target.side.pokemon) {

if (pokemon === target) continue;

for (const moveSlot of pokemon.moveSlots) {

const moveid = moveSlot.id;

if (noAssist.includes(moveid)) continue;

const move = this.dex.getMove(moveid);

if (move.isZ || move.isMax) {

continue;

}

moves.push(moveid);

}

}

let randomMove = '';

if (moves.length) randomMove = this.sample(moves);

if (!randomMove) {

return false;

}

this.useMove(randomMove, target);

},

secondary: null,

target: "self",

type: "Normal",

contestType: "Cute",

},

assurance: {

num: 372,

accuracy: 100,

basePower: 60,

basePowerCallback(pokemon, target, move) {

if (target.hurtThisTurn) {

this.debug('Boosted for being damaged this turn');

return move.basePower \* 2;

}

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the target has already taken damage this turn, other than direct damage from Belly Drum, confusion, Curse, or Pain Split.",

shortDesc: "Power doubles if target was damaged this turn.",

name: "Assurance",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

astonish: {

num: 310,

accuracy: 100,

basePower: 30,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Astonish",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Ghost",

contestType: "Cute",

},

attackorder: {

num: 454,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Attack Order",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Bug",

contestType: "Clever",

},

attract: {

num: 213,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to become infatuated, making it unable to attack 50% of the time. Fails if both the user and the target are the same gender, if either is genderless, or if the target is already infatuated. The effect ends when either the user or the target is no longer active. Pokemon with the Oblivious Ability or protected by the Aroma Veil Ability are immune.",

shortDesc: "A target of the opposite gender gets infatuated.",

name: "Attract",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'attract',

condition: {

noCopy: true, // doesn't get copied by Baton Pass

onStart(pokemon, source, effect) {

if (!(pokemon.gender === 'M' && source.gender === 'F') && !(pokemon.gender === 'F' && source.gender === 'M')) {

this.debug('incompatible gender');

return false;

}

if (!this.runEvent('Attract', pokemon, source)) {

this.debug('Attract event failed');

return false;

}

if (effect.id === 'cutecharm') {

this.add('-start', pokemon, 'Attract', '[from] ability: Cute Charm', '[of] ' + source);

} else if (effect.id === 'destinyknot') {

this.add('-start', pokemon, 'Attract', '[from] item: Destiny Knot', '[of] ' + source);

} else {

this.add('-start', pokemon, 'Attract');

}

},

onUpdate(pokemon) {

if (this.effectData.source && !this.effectData.source.isActive && pokemon.volatiles['attract']) {

this.debug('Removing Attract volatile on ' + pokemon);

pokemon.removeVolatile('attract');

}

},

onBeforeMovePriority: 2,

onBeforeMove(pokemon, target, move) {

this.add('-activate', pokemon, 'move: Attract', '[of] ' + this.effectData.source);

if (this.randomChance(1, 2)) {

this.add('cant', pokemon, 'Attract');

return false;

}

},

onEnd(pokemon) {

this.add('-end', pokemon, 'Attract', '[silent]');

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

aurasphere: {

num: 396,

accuracy: true,

basePower: 80,

category: "Special",

shortDesc: "This move does not check accuracy.",

name: "Aura Sphere",

pp: 20,

priority: 0,

flags: {bullet: 1, protect: 1, pulse: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Fighting",

contestType: "Beautiful",

},

aurawheel: {

num: 783,

accuracy: 100,

basePower: 110,

category: "Physical",

desc: "Has a 100% chance to raise the user's Speed by 1 stage. If the user is a Morpeko in Full Belly Mode, this move is Electric type. If the user is a Morpeko in Hangry Mode, this move is Dark type. This move cannot be used successfully unless the user's current form, while considering Transform, is Full Belly or Hangry Mode Morpeko.",

shortDesc: "Morpeko: Electric; Hangry: Dark; 100% +1 Spe.",

name: "Aura Wheel",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

self: {

boosts: {

spe: 1,

},

},

},

onTry(pokemon) {

if (pokemon.species.baseSpecies === 'Morpeko') {

return;

}

this.hint("Only a Pokemon whose form is Morpeko or Morpeko-Hangry can use this move.");

this.add('-fail', pokemon, 'move: Aura Wheel');

return null;

},

onModifyType(move, pokemon) {

if (pokemon.species.name === 'Morpeko-Hangry') {

move.type = 'Dark';

} else {

move.type = 'Electric';

}

},

target: "normal",

type: "Electric",

},

aurorabeam: {

num: 62,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Has a 10% chance to lower the target's Attack by 1 stage.",

shortDesc: "10% chance to lower the target's Attack by 1.",

name: "Aurora Beam",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

atk: -1,

},

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

auroraveil: {

num: 694,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members take 0.5x damage from physical and special attacks, or 0.66x damage if in a Double Battle; does not reduce damage further with Reflect or Light Screen. Critical hits ignore this protection. It is removed from the user's side if the user or an ally is successfully hit by Brick Break, Psychic Fangs, or Defog. Brick Break and Psychic Fangs remove the effect before damage is calculated. Lasts for 8 turns if the user is holding Light Clay. Fails unless the weather is Hail.",

shortDesc: "For 5 turns, damage to allies is halved. Hail only.",

name: "Aurora Veil",

pp: 20,

priority: 0,

flags: {snatch: 1},

sideCondition: 'auroraveil',

onTryHitSide() {

if (!this.field.isWeather('hail')) return false;

},

condition: {

duration: 5,

durationCallback(target, source, effect) {

if (source?.hasItem('lightclay')) {

return 8;

}

return 5;

},

onAnyModifyDamage(damage, source, target, move) {

if (target !== source && target.side === this.effectData.target) {

if ((target.side.getSideCondition('reflect') && this.getCategory(move) === 'Physical') ||

(target.side.getSideCondition('lightscreen') && this.getCategory(move) === 'Special')) {

return;

}

if (!target.getMoveHitData(move).crit && !move.infiltrates) {

this.debug('Aurora Veil weaken');

if (target.side.active.length > 1) return this.chainModify([0xAAC, 0x1000]);

return this.chainModify(0.5);

}

}

},

onStart(side) {

this.add('-sidestart', side, 'move: Aurora Veil');

},

onResidualOrder: 21,

onResidualSubOrder: 1,

onEnd(side) {

this.add('-sideend', side, 'move: Aurora Veil');

},

},

secondary: null,

target: "allySide",

type: "Ice",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

autotomize: {

num: 475,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Speed by 2 stages. If the user's Speed was changed, the user's weight is reduced by 100 kg as long as it remains active. This effect is stackable but cannot reduce the user's weight to less than 0.1 kg.",

shortDesc: "Raises the user's Speed by 2; user loses 100 kg.",

name: "Autotomize",

pp: 15,

priority: 0,

flags: {snatch: 1},

onTryHit(pokemon) {

const hasContrary = pokemon.hasAbility('contrary');

if ((!hasContrary && pokemon.boosts.spe === 6) || (hasContrary && pokemon.boosts.spe === -6)) {

return false;

}

},

boosts: {

spe: 2,

},

onHit(pokemon) {

if (pokemon.weighthg > 1) {

pokemon.weighthg = Math.max(1, pokemon.weighthg - 1000);

this.add('-start', pokemon, 'Autotomize');

}

},

secondary: null,

target: "self",

type: "Steel",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

avalanche: {

num: 419,

accuracy: 100,

basePower: 60,

basePowerCallback(pokemon, target, move) {

const damagedByTarget = pokemon.attackedBy.some(

p => p.source === target && p.damage > 0 && p.thisTurn

);

if (damagedByTarget) {

this.debug('Boosted for getting hit by ' + target);

return move.basePower \* 2;

}

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the user was hit by the target this turn.",

shortDesc: "Power doubles if user is damaged by the target.",

name: "Avalanche",

pp: 10,

priority: -4,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

babydolleyes: {

num: 608,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 1 stage.",

shortDesc: "Lowers the target's Attack by 1.",

name: "Baby-Doll Eyes",

pp: 30,

priority: 1,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

boosts: {

atk: -1,

},

secondary: null,

target: "normal",

type: "Fairy",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

baddybad: {

num: 737,

accuracy: 95,

basePower: 80,

category: "Special",

desc: "This move summons Reflect for 5 turns upon use.",

shortDesc: "Summons Reflect.",

isNonstandard: "LGPE",

name: "Baddy Bad",

pp: 15,

priority: 0,

flags: {protect: 1},

self: {

sideCondition: 'reflect',

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

banefulbunker: {

num: 661,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn, and Pokemon making contact with the user become poisoned. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Protects from moves. Contact: poison.",

name: "Baneful Bunker",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'banefulbunker',

onTryHit(target, source, move) {

return !!this.queue.willAct() && this.runEvent('StallMove', target);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'move: Protect');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect']) {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Protect');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

if (move.flags['contact']) {

source.trySetStatus('psn', target);

}

return this.NOT\_FAIL;

},

onHit(target, source, move) {

if (move.isZOrMaxPowered && move.flags['contact']) {

source.trySetStatus('psn', target);

}

},

},

secondary: null,

target: "self",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

barrage: {

num: 140,

accuracy: 85,

basePower: 15,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

isNonstandard: "Past",

name: "Barrage",

pp: 20,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

barrier: {

num: 112,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 2 stages.",

shortDesc: "Raises the user's Defense by 2.",

isNonstandard: "Past",

name: "Barrier",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 2,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cool",

},

batonpass: {

num: 226,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is replaced with another Pokemon in its party. The selected Pokemon has the user's stat stage changes, confusion, and certain move effects transferred to it.",

shortDesc: "User switches, passing stat changes and more.",

name: "Baton Pass",

pp: 40,

priority: 0,

flags: {},

selfSwitch: 'copyvolatile',

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

beakblast: {

num: 690,

accuracy: 100,

basePower: 100,

category: "Physical",

desc: "If the user is hit by a contact move this turn before it can execute this move, the attacker is burned.",

shortDesc: "Burns on contact with the user before it moves.",

isNonstandard: "Past",

name: "Beak Blast",

pp: 15,

priority: -3,

flags: {bullet: 1, protect: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('beakblast');

},

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'move: Beak Blast');

},

onHit(pokemon, source, move) {

if (move.flags['contact']) {

source.trySetStatus('brn', pokemon);

}

},

},

// FIXME: onMoveAborted(pokemon) {pokemon.removeVolatile('beakblast')},

onAfterMove(pokemon) {

pokemon.removeVolatile('beakblast');

},

secondary: null,

target: "normal",

type: "Flying",

contestType: "Tough",

},

beatup: {

num: 251,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target, move) {

return 5 + Math.floor(move.allies!.shift()!.species.baseStats.atk / 10);

},

category: "Physical",

desc: "Hits one time for the user and one time for each unfainted Pokemon without a major status condition in the user's party. The power of each hit is equal to 5+(X/10), where X is each participating Pokemon's base Attack; each hit is considered to come from the user.",

shortDesc: "All healthy allies aid in damaging the target.",

name: "Beat Up",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

onModifyMove(move, pokemon) {

move.allies = pokemon.side.pokemon.filter(ally => ally === pokemon || !ally.fainted && !ally.status);

move.multihit = move.allies.length;

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

behemothbash: {

num: 782,

accuracy: 100,

basePower: 100,

category: "Physical",

shortDesc: "Damage doubles if the target is Dynamaxed.",

name: "Behemoth Bash",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

},

behemothblade: {

num: 781,

accuracy: 100,

basePower: 100,

category: "Physical",

shortDesc: "Damage doubles if the target is Dynamaxed.",

name: "Behemoth Blade",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

},

belch: {

num: 562,

accuracy: 90,

basePower: 120,

category: "Special",

desc: "This move cannot be selected until the user eats a Berry, either by eating one that was held, stealing and eating one off another Pokemon with Bug Bite or Pluck, or eating one that was thrown at it with Fling. Once the condition is met, this move can be selected and used for the rest of the battle even if the user gains or uses another item or switches out. Consuming a Berry with Natural Gift does not count for the purposes of eating one.",

shortDesc: "Cannot be selected until the user eats a Berry.",

name: "Belch",

pp: 10,

priority: 0,

flags: {protect: 1},

// Move disabling implemented in Battle#nextTurn in sim/battle.js

secondary: null,

target: "normal",

type: "Poison",

contestType: "Tough",

},

bellydrum: {

num: 187,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack by 12 stages in exchange for the user losing 1/2 of its maximum HP, rounded down. Fails if the user would faint or if its Attack stat stage is 6.",

shortDesc: "User loses 50% max HP. Maximizes Attack.",

name: "Belly Drum",

pp: 10,

priority: 0,

flags: {snatch: 1},

onHit(target) {

if (target.hp <= target.maxhp / 2 || target.boosts.atk >= 6 || target.maxhp === 1) { // Shedinja clause

return false;

}

this.directDamage(target.maxhp / 2);

this.boost({atk: 12}, target);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Cute",

},

bestow: {

num: 516,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target receives the user's held item. Fails if the user has no item or is holding a Mail or Z-Crystal, if the target is already holding an item, if the user is a Kyogre holding a Blue Orb, a Groudon holding a Red Orb, a Giratina holding a Griseous Orb, an Arceus holding a Plate, a Genesect holding a Drive, a Silvally holding a Memory, a Pokemon that can Mega Evolve holding the Mega Stone for its species, or if the target is one of those Pokemon and the user is holding the respective item.",

shortDesc: "User passes its held item to the target.",

isNonstandard: "Past",

name: "Bestow",

pp: 15,

priority: 0,

flags: {mirror: 1, authentic: 1, mystery: 1},

onHit(target, source, move) {

if (target.item) {

return false;

}

const myItem = source.takeItem();

if (!myItem) return false;

if (!this.singleEvent('TakeItem', myItem, source.itemData, target, source, move, myItem) || !target.setItem(myItem)) {

source.item = myItem.id;

return false;

}

this.add('-item', target, myItem.name, '[from] move: Bestow', '[of] ' + source);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 2}},

contestType: "Cute",

},

bide: {

num: 117,

accuracy: true,

basePower: 0,

category: "Physical",

desc: "The user spends two turns locked into this move and then, on the second turn after using this move, the user attacks the last Pokemon that hit it, inflicting double the damage in HP it lost to attacks during the two turns. If the last Pokemon that hit it is no longer active, the user attacks a random opposing Pokemon instead. If the user is prevented from moving during this move's use, the effect ends. This move does not check accuracy and does not ignore type immunity.",

shortDesc: "Waits 2 turns; deals double the damage taken.",

isNonstandard: "Past",

name: "Bide",

pp: 10,

priority: 1,

flags: {contact: 1, protect: 1},

volatileStatus: 'bide',

ignoreImmunity: true,

beforeMoveCallback(pokemon) {

if (pokemon.volatiles['bide']) return true;

},

condition: {

duration: 3,

onLockMove: 'bide',

onStart(pokemon) {

this.effectData.totalDamage = 0;

this.add('-start', pokemon, 'move: Bide');

},

onDamagePriority: -101,

onDamage(damage, target, source, move) {

if (!move || move.effectType !== 'Move' || !source) return;

this.effectData.totalDamage += damage;

this.effectData.lastDamageSource = source;

},

onBeforeMove(pokemon, target, move) {

if (this.effectData.duration === 1) {

this.add('-end', pokemon, 'move: Bide');

target = this.effectData.lastDamageSource;

if (!target || !this.effectData.totalDamage) {

this.attrLastMove('[still]');

this.add('-fail', pokemon);

return false;

}

if (!target.isActive) {

const possibleTarget = this.getRandomTarget(pokemon, this.dex.getMove('pound'));

if (!possibleTarget) {

this.add('-miss', pokemon);

return false;

}

target = possibleTarget;

}

const moveData: Partial<ActiveMove> = {

id: 'bide' as ID,

name: "Bide",

accuracy: true,

damage: this.effectData.totalDamage \* 2,

category: "Physical",

priority: 1,

flags: {contact: 1, protect: 1},

effectType: 'Move',

type: 'Normal',

};

this.tryMoveHit(target, pokemon, moveData as ActiveMove);

return false;

}

this.add('-activate', pokemon, 'move: Bide');

},

onMoveAborted(pokemon) {

pokemon.removeVolatile('bide');

},

onEnd(pokemon) {

this.add('-end', pokemon, 'move: Bide', '[silent]');

},

},

secondary: null,

target: "self",

type: "Normal",

contestType: "Tough",

},

bind: {

num: 20,

accuracy: 85,

basePower: 15,

category: "Physical",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Bind",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

bite: {

num: 44,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Bite",

pp: 25,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Dark",

contestType: "Tough",

},

blackholeeclipse: {

num: 654,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Black Hole Eclipse",

pp: 1,

priority: 0,

flags: {},

isZ: "darkiniumz",

secondary: null,

target: "normal",

type: "Dark",

contestType: "Cool",

},

blastburn: {

num: 307,

accuracy: 90,

basePower: 150,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Blast Burn",

pp: 5,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

blazekick: {

num: 299,

accuracy: 90,

basePower: 85,

category: "Physical",

desc: "Has a 10% chance to burn the target and a higher chance for a critical hit.",

shortDesc: "High critical hit ratio. 10% chance to burn.",

name: "Blaze Kick",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Cool",

},

blizzard: {

num: 59,

accuracy: 70,

basePower: 110,

category: "Special",

desc: "Has a 10% chance to freeze the target. If the weather is Hail, this move does not check accuracy.",

shortDesc: "10% chance to freeze foe(s). Can't miss in hail.",

name: "Blizzard",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move) {

if (this.field.isWeather('hail')) move.accuracy = true;

},

secondary: {

chance: 10,

status: 'frz',

},

target: "allAdjacentFoes",

type: "Ice",

contestType: "Beautiful",

},

block: {

num: 335,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Prevents the target from switching out.",

name: "Block",

pp: 5,

priority: 0,

flags: {reflectable: 1, mirror: 1},

onHit(target, source, move) {

return target.addVolatile('trapped', source, move, 'trapper');

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

bloomdoom: {

num: 644,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Bloom Doom",

pp: 1,

priority: 0,

flags: {},

isZ: "grassiumz",

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

blueflare: {

num: 551,

accuracy: 85,

basePower: 130,

category: "Special",

desc: "Has a 20% chance to burn the target.",

shortDesc: "20% chance to burn the target.",

name: "Blue Flare",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

bodypress: {

num: 776,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Damage is calculated using the user's Defense stat as its Attack, including stat stage changes. Other effects that modify the Attack stat are used as normal.",

shortDesc: "Uses user's Def stat as Atk in damage calculation.",

name: "Body Press",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

useSourceDefensiveAsOffensive: true,

secondary: null,

target: "normal",

type: "Fighting",

},

bodyslam: {

num: 34,

accuracy: 100,

basePower: 85,

category: "Physical",

desc: "Has a 30% chance to paralyze the target. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "30% chance to paralyze the target.",

name: "Body Slam",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Normal",

contestType: "Tough",

},

boltbeak: {

num: 754,

accuracy: 100,

basePower: 85,

basePowerCallback(pokemon, target, move) {

if (target.newlySwitched || this.queue.willMove(target)) {

this.debug('Bolt Beak damage boost');

return move.basePower \* 2;

}

this.debug('Bolt Beak NOT boosted');

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the user moves before the target.",

shortDesc: "Power doubles if user moves before the target.",

name: "Bolt Beak",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Electric",

},

boltstrike: {

num: 550,

accuracy: 85,

basePower: 130,

category: "Physical",

desc: "Has a 20% chance to paralyze the target.",

shortDesc: "20% chance to paralyze the target.",

name: "Bolt Strike",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Beautiful",

},

boneclub: {

num: 125,

accuracy: 85,

basePower: 65,

category: "Physical",

desc: "Has a 10% chance to flinch the target.",

shortDesc: "10% chance to flinch the target.",

isNonstandard: "Past",

name: "Bone Club",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'flinch',

},

target: "normal",

type: "Ground",

contestType: "Tough",

},

bonemerang: {

num: 155,

accuracy: 90,

basePower: 50,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Bonemerang",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Ground",

maxMove: {basePower: 130},

contestType: "Tough",

},

bonerush: {

num: 198,

accuracy: 90,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Bone Rush",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Ground",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Tough",

},

boomburst: {

num: 586,

accuracy: 100,

basePower: 140,

category: "Special",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent Pokemon.",

name: "Boomburst",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: null,

target: "allAdjacent",

type: "Normal",

contestType: "Tough",

},

bounce: {

num: 340,

accuracy: 85,

basePower: 85,

category: "Physical",

desc: "Has a 30% chance to paralyze the target. This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks other than Gust, Hurricane, Sky Uppercut, Smack Down, Thousand Arrows, Thunder, and Twister, and Gust and Twister have doubled power when used against it. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Bounces turn 1. Hits turn 2. 30% paralyze.",

name: "Bounce",

pp: 5,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1, gravity: 1, distance: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onInvulnerability(target, source, move) {

if (['gust', 'twister', 'skyuppercut', 'thunder', 'hurricane', 'smackdown', 'thousandarrows'].includes(move.id)) {

return;

}

return false;

},

onSourceBasePower(basePower, target, source, move) {

if (move.id === 'gust' || move.id === 'twister') {

return this.chainModify(2);

}

},

},

secondary: {

chance: 30,

status: 'par',

},

target: "any",

type: "Flying",

contestType: "Cute",

},

bouncybubble: {

num: 733,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

isNonstandard: "LGPE",

name: "Bouncy Bubble",

pp: 20,

priority: 0,

flags: {protect: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Water",

contestType: "Clever",

},

branchpoke: {

num: 785,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "No additional effect.",

name: "Branch Poke",

pp: 40,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

},

bravebird: {

num: 413,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 33% the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 33% recoil.",

name: "Brave Bird",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

recoil: [33, 100],

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

breakingswipe: {

num: 784,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 100% chance to lower the target's Attack by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Attack by 1.",

name: "Breaking Swipe",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

atk: -1,

},

},

target: "allAdjacentFoes",

type: "Dragon",

},

breakneckblitz: {

num: 622,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Breakneck Blitz",

pp: 1,

priority: 0,

flags: {},

isZ: "normaliumz",

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

brickbreak: {

num: 280,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "If this attack does not miss, the effects of Reflect, Light Screen, and Aurora Veil end for the target's side of the field before damage is calculated.",

shortDesc: "Destroys screens, unless the target is immune.",

name: "Brick Break",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onTryHit(pokemon) {

// will shatter screens through sub, before you hit

if (pokemon.runImmunity('Fighting')) {

pokemon.side.removeSideCondition('reflect');

pokemon.side.removeSideCondition('lightscreen');

pokemon.side.removeSideCondition('auroraveil');

}

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

brine: {

num: 362,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Power doubles if the target has less than or equal to half of its maximum HP remaining.",

shortDesc: "Power doubles if the target's HP is 50% or less.",

name: "Brine",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower, pokemon, target) {

if (target.hp \* 2 <= target.maxhp) {

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Water",

contestType: "Tough",

},

brutalswing: {

num: 693,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent Pokemon.",

name: "Brutal Swing",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "allAdjacent",

type: "Dark",

contestType: "Tough",

},

bubble: {

num: 145,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 10% chance to lower the target's Speed by 1 stage.",

shortDesc: "10% chance to lower the foe(s) Speed by 1.",

isNonstandard: "Past",

name: "Bubble",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spe: -1,

},

},

target: "allAdjacentFoes",

type: "Water",

contestType: "Cute",

},

bubblebeam: {

num: 61,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Has a 10% chance to lower the target's Speed by 1 stage.",

shortDesc: "10% chance to lower the target's Speed by 1.",

name: "Bubble Beam",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Water",

contestType: "Beautiful",

},

bugbite: {

num: 450,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "If this move is successful and the user has not fainted, it steals the target's held Berry if it is holding one and eats it immediately, gaining its effects even if the user's item is being ignored. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "User steals and eats the target's Berry.",

name: "Bug Bite",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onHit(target, source) {

const item = target.getItem();

if (source.hp && item.isBerry && target.takeItem(source)) {

this.add('-enditem', target, item.name, '[from] stealeat', '[move] Bug Bite', '[of] ' + source);

if (this.singleEvent('Eat', item, null, source, null, null)) {

this.runEvent('EatItem', source, null, null, item);

if (item.id === 'leppaberry') target.staleness = 'external';

}

if (item.onEat) source.ateBerry = true;

}

},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cute",

},

bugbuzz: {

num: 405,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Bug Buzz",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Bug",

contestType: "Beautiful",

},

bulkup: {

num: 339,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack and Defense by 1 stage.",

shortDesc: "Raises the user's Attack and Defense by 1.",

name: "Bulk Up",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

def: 1,

},

secondary: null,

target: "self",

type: "Fighting",

zMove: {boost: {atk: 1}},

contestType: "Cool",

},

bulldoze: {

num: 523,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance lower adjacent Pkmn Speed by 1.",

name: "Bulldoze",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "allAdjacent",

type: "Ground",

contestType: "Tough",

},

bulletpunch: {

num: 418,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Bullet Punch",

pp: 30,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: null,

target: "normal",

type: "Steel",

contestType: "Tough",

},

bulletseed: {

num: 331,

accuracy: 100,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Bullet Seed",

pp: 30,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Grass",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Cool",

},

burningjealousy: {

num: 807,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "Has a 100% chance to burn the target if it had a stat stage raised this turn.",

shortDesc: "100% burns a target that had a stat rise this turn.",

name: "Burning Jealousy",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

onHit(target, source, move) {

if (target?.statsRaisedThisTurn) {

target.trySetStatus('brn', source, move);

}

},

},

target: "allAdjacentFoes",

type: "Fire",

contestType: "Tough",

},

burnup: {

num: 682,

accuracy: 100,

basePower: 130,

category: "Special",

desc: "Fails unless the user is a Fire type. If this move is successful, the user's Fire type becomes typeless as long as it remains active.",

shortDesc: "User's Fire type becomes typeless; must be Fire.",

name: "Burn Up",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

onTryMove(pokemon, target, move) {

if (pokemon.hasType('Fire')) return;

this.add('-fail', pokemon, 'move: Burn Up');

this.attrLastMove('[still]');

return null;

},

self: {

onHit(pokemon) {

pokemon.setType(pokemon.getTypes(true).map(type => type === "Fire" ? "???" : type));

this.add('-start', pokemon, 'typechange', pokemon.types.join('/'), '[from] move: Burn Up');

},

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Clever",

},

buzzybuzz: {

num: 734,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 100% chance to paralyze the foe.",

shortDesc: "100% chance to paralyze the foe.",

isNonstandard: "LGPE",

name: "Buzzy Buzz",

pp: 20,

priority: 0,

flags: {protect: 1},

secondary: {

chance: 100,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Clever",

},

calmmind: {

num: 347,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Attack and Special Defense by 1 stage.",

shortDesc: "Raises the user's Sp. Atk and Sp. Def by 1.",

name: "Calm Mind",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

spa: 1,

spd: 1,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

camouflage: {

num: 293,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user's type changes based on the battle terrain. Normal type on the regular Wi-Fi terrain, Electric type during Electric Terrain, Fairy type during Misty Terrain, Grass type during Grassy Terrain, and Psychic type during Psychic Terrain. Fails if the user's type cannot be changed or if the user is already purely that type.",

shortDesc: "Changes user's type by terrain (default Normal).",

isNonstandard: "Past",

name: "Camouflage",

pp: 20,

priority: 0,

flags: {snatch: 1},

onHit(target) {

let newType = 'Normal';

if (this.field.isTerrain('electricterrain')) {

newType = 'Electric';

} else if (this.field.isTerrain('grassyterrain')) {

newType = 'Grass';

} else if (this.field.isTerrain('mistyterrain')) {

newType = 'Fairy';

} else if (this.field.isTerrain('psychicterrain')) {

newType = 'Psychic';

}

if (target.getTypes().join() === newType || !target.setType(newType)) return false;

this.add('-start', target, 'typechange', newType);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {evasion: 1}},

contestType: "Clever",

},

captivate: {

num: 445,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Special Attack by 2 stages. The target is unaffected if both the user and the target are the same gender, or if either is genderless. Pokemon with the Oblivious Ability are immune.",

shortDesc: "Lowers the foe(s) Sp. Atk by 2 if opposite gender.",

isNonstandard: "Past",

name: "Captivate",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

onTryImmunity(pokemon, source) {

return (pokemon.gender === 'M' && source.gender === 'F') || (pokemon.gender === 'F' && source.gender === 'M');

},

boosts: {

spa: -2,

},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

zMove: {boost: {spd: 2}},

contestType: "Cute",

},

catastropika: {

num: 658,

accuracy: true,

basePower: 210,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Catastropika",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "pikaniumz",

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

celebrate: {

num: 606,

accuracy: true,

basePower: 0,

category: "Status",

shortDesc: "No competitive use.",

name: "Celebrate",

pp: 40,

priority: 0,

flags: {},

onTryHit(target, source) {

this.add('-activate', target, 'move: Celebrate');

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Cute",

},

charge: {

num: 268,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Defense by 1 stage. If the user uses an Electric-type attack on the next turn, its power will be doubled.",

shortDesc: "+1 SpD, user's Electric move next turn 2x power.",

name: "Charge",

pp: 20,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'charge',

onHit(pokemon) {

this.add('-activate', pokemon, 'move: Charge');

},

condition: {

duration: 2,

onRestart(pokemon) {

this.effectData.duration = 2;

},

onBasePowerPriority: 9,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Electric') {

this.debug('charge boost');

return this.chainModify(2);

}

},

},

boosts: {

spd: 1,

},

secondary: null,

target: "self",

type: "Electric",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

chargebeam: {

num: 451,

accuracy: 90,

basePower: 50,

category: "Special",

desc: "Has a 70% chance to raise the user's Special Attack by 1 stage.",

shortDesc: "70% chance to raise the user's Sp. Atk by 1.",

name: "Charge Beam",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 70,

self: {

boosts: {

spa: 1,

},

},

},

target: "normal",

type: "Electric",

contestType: "Beautiful",

},

charm: {

num: 204,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 2 stages.",

shortDesc: "Lowers the target's Attack by 2.",

name: "Charm",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

boosts: {

atk: -2,

},

secondary: null,

target: "normal",

type: "Fairy",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

chatter: {

num: 448,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Has a 100% chance to confuse the target.",

shortDesc: "100% chance to confuse the target.",

isNonstandard: "Past",

name: "Chatter",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, distance: 1, authentic: 1},

noSketch: true,

secondary: {

chance: 100,

volatileStatus: 'confusion',

},

target: "any",

type: "Flying",

contestType: "Cute",

},

chipaway: {

num: 498,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Ignores the target's stat stage changes, including evasiveness.",

shortDesc: "Ignores the target's stat stage changes.",

isNonstandard: "Past",

name: "Chip Away",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ignoreDefensive: true,

ignoreEvasion: true,

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

circlethrow: {

num: 509,

accuracy: 90,

basePower: 60,

category: "Physical",

desc: "If both the user and the target have not fainted, the target is forced to switch out and be replaced with a random unfainted ally. This effect fails if the target is under the effect of Ingrain, has the Suction Cups Ability, or this move hit a substitute.",

shortDesc: "Forces the target to switch to a random ally.",

name: "Circle Throw",

pp: 10,

priority: -6,

flags: {contact: 1, protect: 1, mirror: 1},

forceSwitch: true,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

clamp: {

num: 128,

accuracy: 85,

basePower: 35,

category: "Physical",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

isNonstandard: "Past",

name: "Clamp",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Water",

contestType: "Tough",

},

clangingscales: {

num: 691,

accuracy: 100,

basePower: 110,

category: "Special",

desc: "Lowers the user's Defense by 1 stage.",

shortDesc: "Lowers the user's Defense by 1.",

name: "Clanging Scales",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

selfBoost: {

boosts: {

def: -1,

},

},

secondary: null,

target: "allAdjacentFoes",

type: "Dragon",

contestType: "Tough",

},

clangoroussoul: {

num: 775,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage in exchange for the user losing 33% of its maximum HP, rounded down. Fails if the user would faint or if its Attack, Defense, Special Attack, Special Defense, and Speed stat stages would not change.",

shortDesc: "User loses 33% of its max HP. +1 to all stats.",

name: "Clangorous Soul",

pp: 5,

priority: 0,

flags: {snatch: 1, sound: 1, dance: 1},

onTryHit(pokemon, target, move) {

if (pokemon.hp <= (pokemon.maxhp \* 33 / 100) || pokemon.maxhp === 1) {

return false;

}

if (!this.boost(move.boosts as SparseBoostsTable)) return null;

delete move.boosts;

},

onHit(pokemon) {

this.directDamage(pokemon.maxhp \* 33 / 100);

},

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

secondary: null,

target: "self",

type: "Dragon",

},

clangoroussoulblaze: {

num: 728,

accuracy: true,

basePower: 185,

category: "Special",

desc: "Raises the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage.",

shortDesc: "Raises the user's Atk/Def/SpAtk/SpDef/Spe by 1.",

isNonstandard: "Past",

name: "Clangorous Soulblaze",

pp: 1,

priority: 0,

flags: {sound: 1, authentic: 1},

selfBoost: {

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

},

isZ: "kommoniumz",

secondary: {

// Sheer Force negates the selfBoost even though it is not secondary

},

target: "allAdjacentFoes",

type: "Dragon",

contestType: "Cool",

},

clearsmog: {

num: 499,

accuracy: true,

basePower: 50,

category: "Special",

shortDesc: "Resets all of the target's stat stages to 0.",

name: "Clear Smog",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit(target) {

target.clearBoosts();

this.add('-clearboost', target);

},

secondary: null,

target: "normal",

type: "Poison",

contestType: "Beautiful",

},

closecombat: {

num: 370,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "Lowers the user's Defense and Special Defense by 1 stage.",

shortDesc: "Lowers the user's Defense and Sp. Def by 1.",

name: "Close Combat",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

boosts: {

def: -1,

spd: -1,

},

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

coaching: {

num: 811,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the target's Attack and Defense by 1 stage. Fails if there is no ally adjacent to the user.",

shortDesc: "Raises an ally's Attack and Defense by 1.",

name: "Coaching",

pp: 10,

priority: 0,

flags: {authentic: 1},

secondary: null,

boosts: {

atk: 1,

def: 1,

},

target: "adjacentAlly",

type: "Fighting",

},

coil: {

num: 489,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack, Defense, and accuracy by 1 stage.",

shortDesc: "Raises user's Attack, Defense, accuracy by 1.",

name: "Coil",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

def: 1,

accuracy: 1,

},

secondary: null,

target: "self",

type: "Poison",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

cometpunch: {

num: 4,

accuracy: 85,

basePower: 18,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

isNonstandard: "Past",

name: "Comet Punch",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

maxMove: {basePower: 100},

contestType: "Tough",

},

confide: {

num: 590,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Lowers the target's Special Attack by 1 stage.",

shortDesc: "Lowers the target's Sp. Atk by 1.",

name: "Confide",

pp: 20,

priority: 0,

flags: {reflectable: 1, mirror: 1, sound: 1, authentic: 1},

boosts: {

spa: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

confuseray: {

num: 109,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to become confused.",

shortDesc: "Confuses the target.",

name: "Confuse Ray",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'confusion',

secondary: null,

target: "normal",

type: "Ghost",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

confusion: {

num: 93,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "Has a 10% chance to confuse the target.",

shortDesc: "10% chance to confuse the target.",

name: "Confusion",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'confusion',

},

target: "normal",

type: "Psychic",

contestType: "Clever",

},

constrict: {

num: 132,

accuracy: 100,

basePower: 10,

category: "Physical",

desc: "Has a 10% chance to lower the target's Speed by 1 stage.",

shortDesc: "10% chance to lower the target's Speed by 1.",

isNonstandard: "Past",

name: "Constrict",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Normal",

contestType: "Tough",

},

continentalcrush: {

num: 632,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Continental Crush",

pp: 1,

priority: 0,

flags: {},

isZ: "rockiumz",

secondary: null,

target: "normal",

type: "Rock",

contestType: "Cool",

},

conversion: {

num: 160,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user's type changes to match the original type of the move in its first move slot. Fails if the user cannot change its type, or if the type is one of the user's current types.",

shortDesc: "Changes user's type to match its first move.",

name: "Conversion",

pp: 30,

priority: 0,

flags: {snatch: 1},

onHit(target) {

const type = this.dex.getMove(target.moveSlots[0].id).type;

if (target.hasType(type) || !target.setType(type)) return false;

this.add('-start', target, 'typechange', type);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Beautiful",

},

conversion2: {

num: 176,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user's type changes to match a type that resists or is immune to the type of the last move used by the target, but not either of its current types. The determined type of the move is used rather than the original type. Fails if the target has not made a move, if the user cannot change its type, or if this move would only be able to select one of the user's current types.",

shortDesc: "Changes user's type to resist target's last move.",

name: "Conversion 2",

pp: 30,

priority: 0,

flags: {authentic: 1},

onHit(target, source) {

if (!target.lastMove) {

return false;

}

const possibleTypes = [];

const attackType = target.lastMove.type;

for (const type in this.dex.data.TypeChart) {

if (source.hasType(type)) continue;

const typeCheck = this.dex.data.TypeChart[type].damageTaken[attackType];

if (typeCheck === 2 || typeCheck === 3) {

possibleTypes.push(type);

}

}

if (!possibleTypes.length) {

return false;

}

const randomType = this.sample(possibleTypes);

if (!source.setType(randomType)) return false;

this.add('-start', source, 'typechange', randomType);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Beautiful",

},

copycat: {

num: 383,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user uses the last move used by any Pokemon, including itself. The base move of Max and G-Max Moves is considered for this purpose. Fails if no move has been used, or if the last move used was Assist, Baneful Bunker, Beak Blast, Belch, Bestow, Celebrate, Chatter, Circle Throw, Copycat, Counter, Covet, Crafty Shield, Destiny Bond, Detect, Dragon Tail, Dynamax Cannon, Endure, Feint, Focus Punch, Follow Me, Helping Hand, Hold Hands, King's Shield, Mat Block, Me First, Metronome, Mimic, Mirror Coat, Mirror Move, Nature Power, Obstruct, Protect, Rage Powder, Roar, Shell Trap, Sketch, Sleep Talk, Snatch, Spiky Shield, Spotlight, Struggle, Switcheroo, Thief, Transform, Trick, or Whirlwind.",

shortDesc: "Uses the last move used in the battle.",

name: "Copycat",

pp: 20,

priority: 0,

flags: {},

onHit(pokemon) {

const noCopycat = [

'assist', 'banefulbunker', 'beakblast', 'belch', 'bestow', 'celebrate', 'chatter', 'circlethrow', 'copycat', 'counter', 'covet', 'craftyshield', 'destinybond', 'detect', 'dragontail', 'dynamaxcannon', 'endure', 'feint', 'focuspunch', 'followme', 'helpinghand', 'holdhands', 'kingsshield', 'matblock', 'mefirst', 'metronome', 'mimic', 'mirrorcoat', 'mirrormove', 'naturepower', 'obstruct', 'protect', 'ragepowder', 'roar', 'shelltrap', 'sketch', 'sleeptalk', 'snatch', 'spikyshield', 'spotlight', 'struggle', 'switcheroo', 'thief', 'transform', 'trick', 'whirlwind',

];

let move: Move | ActiveMove | null = this.lastMove;

if (!move) return;

if (move.isMax && move.baseMove) move = this.dex.getMove(move.baseMove);

if (noCopycat.includes(move.id) || move.isZ || move.isMax) {

return false;

}

this.useMove(move.id, pokemon);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {accuracy: 1}},

contestType: "Cute",

},

coreenforcer: {

num: 687,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "If the user moves after the target, the target's Ability is rendered ineffective as long as it remains active. If the target uses Baton Pass, the replacement will remain under this effect. If the target's Ability is Battle Bond, Comatose, Disguise, Multitype, Power Construct, RKS System, Schooling, Shields Down, Stance Change, or Zen Mode, this effect does not happen, and receiving the effect through Baton Pass ends the effect immediately.",

shortDesc: "Nullifies the foe(s) Ability if the foe(s) move first.",

isNonstandard: "Past",

name: "Core Enforcer",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit(target) {

const noAbilityChange = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'zenmode',

];

if (noAbilityChange.includes(target.ability)) return;

if (target.newlySwitched || this.queue.willMove(target)) return;

target.addVolatile('gastroacid');

},

onAfterSubDamage(damage, target) {

const noAbilityChange = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'zenmode',

];

if (noAbilityChange.includes(target.ability)) return;

if (target.newlySwitched || this.queue.willMove(target)) return;

target.addVolatile('gastroacid');

},

secondary: null,

target: "allAdjacentFoes",

type: "Dragon",

zMove: {basePower: 140},

contestType: "Tough",

},

corkscrewcrash: {

num: 638,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Corkscrew Crash",

pp: 1,

priority: 0,

flags: {},

isZ: "steeliumz",

secondary: null,

target: "normal",

type: "Steel",

contestType: "Cool",

},

corrosivegas: {

num: 810,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "The target loses its held item. This move cannot cause Pokemon with the Sticky Hold Ability to lose their held item or cause a Kyogre, a Groudon, a Giratina, an Arceus, a Genesect, a Silvally, a Zacian, or a Zamazenta to lose their Blue Orb, Red Orb, Griseous Orb, Plate, Drive, Memory, Rusted Sword, or Rusted Shield respectively. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "Removes adjacent Pokemon's held items.",

name: "Corrosive Gas",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target, source) {

const item = target.takeItem(source);

if (item) {

this.add('-enditem', target, item.name, '[from] move: Corrosive Gas', '[of] ' + source);

}

},

secondary: null,

target: "allAdjacent",

type: "Poison",

},

cosmicpower: {

num: 322,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense and Special Defense by 1 stage.",

shortDesc: "Raises the user's Defense and Sp. Def by 1.",

name: "Cosmic Power",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 1,

spd: 1,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {boost: {spd: 1}},

contestType: "Beautiful",

},

cottonguard: {

num: 538,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 3 stages.",

shortDesc: "Raises the user's Defense by 3.",

name: "Cotton Guard",

pp: 10,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 3,

},

secondary: null,

target: "self",

type: "Grass",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

cottonspore: {

num: 178,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Speed by 2 stages.",

shortDesc: "Lowers the target's Speed by 2.",

name: "Cotton Spore",

pp: 40,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1},

boosts: {

spe: -2,

},

secondary: null,

target: "allAdjacentFoes",

type: "Grass",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

counter: {

num: 68,

accuracy: 100,

basePower: 0,

damageCallback(pokemon) {

if (!pokemon.volatiles['counter']) return 0;

return pokemon.volatiles['counter'].damage || 1;

},

category: "Physical",

desc: "Deals damage to the last opposing Pokemon to hit the user with a physical attack this turn equal to twice the HP lost by the user from that attack. If the user did not lose HP from the attack, this move deals 1 HP of damage instead. If that opposing Pokemon's position is no longer in use and there is another opposing Pokemon on the field, the damage is done to it instead. Only the last hit of a multi-hit attack is counted. Fails if the user was not hit by an opposing Pokemon's physical attack this turn.",

shortDesc: "If hit by physical attack, returns double damage.",

name: "Counter",

pp: 20,

priority: -5,

flags: {contact: 1, protect: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('counter');

},

onTryHit(target, source, move) {

if (!source.volatiles['counter']) return false;

if (source.volatiles['counter'].position === null) return false;

},

condition: {

duration: 1,

noCopy: true,

onStart(target, source, move) {

this.effectData.position = null;

this.effectData.damage = 0;

},

onRedirectTargetPriority: -1,

onRedirectTarget(target, source, source2) {

if (source !== this.effectData.target) return;

return source.side.foe.active[this.effectData.position];

},

onDamagingHit(damage, target, source, move) {

if (source.side !== target.side && this.getCategory(move) === 'Physical') {

this.effectData.position = source.position;

this.effectData.damage = 2 \* damage;

}

},

},

secondary: null,

target: "scripted",

type: "Fighting",

maxMove: {basePower: 75},

contestType: "Tough",

},

courtchange: {

num: 756,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Switches the Mist, Light Screen, Reflect, Spikes, Safeguard, Tailwind, Toxic Spikes, Stealth Rock, Water Pledge, Fire Pledge, Grass Pledge, Sticky Web, Aurora Veil, G-Max Steelsurge, G-Max Cannonade, G-Max Vine Lash, and G-Max Wildfire effects from the user's side to the opposing side and vice versa.",

shortDesc: "Swaps user's field effects with the opposing side.",

name: "Court Change",

pp: 10,

priority: 0,

flags: {mirror: 1},

onHitField(target, source) {

const sourceSide = source.side;

const targetSide = source.side.foe;

const sideConditions = [

'mist', 'lightscreen', 'reflect', 'spikes', 'safeguard', 'tailwind', 'toxicspikes', 'stealthrock', 'waterpledge', 'firepledge', 'grasspledge', 'stickyweb', 'auroraveil', 'gmaxsteelsurge', 'gmaxcannonade', 'gmaxvinelash', 'gmaxwildfire',

];

let success = false;

for (const id of sideConditions) {

const effectName = this.dex.getEffect(id).name;

if (sourceSide.sideConditions[id] && targetSide.sideConditions[id]) {

[sourceSide.sideConditions[id], targetSide.sideConditions[id]] = [

targetSide.sideConditions[id], sourceSide.sideConditions[id],

];

this.add('-sideend', sourceSide, effectName, '[silent]');

this.add('-sideend', targetSide, effectName, '[silent]');

} else if (sourceSide.sideConditions[id] && !targetSide.sideConditions[id]) {

targetSide.sideConditions[id] = sourceSide.sideConditions[id];

delete sourceSide.sideConditions[id];

this.add('-sideend', sourceSide, effectName, '[silent]');

} else if (targetSide.sideConditions[id] && !sourceSide.sideConditions[id]) {

sourceSide.sideConditions[id] = targetSide.sideConditions[id];

delete targetSide.sideConditions[id];

this.add('-sideend', targetSide, effectName, '[silent]');

} else {

continue;

}

let sourceLayers = sourceSide.sideConditions[id] ? (sourceSide.sideConditions[id].layers || 1) : 0;

let targetLayers = targetSide.sideConditions[id] ? (targetSide.sideConditions[id].layers || 1) : 0;

for (; sourceLayers > 0; sourceLayers--) {

this.add('-sidestart', sourceSide, effectName, '[silent]');

}

for (; targetLayers > 0; targetLayers--) {

this.add('-sidestart', targetSide, effectName, '[silent]');

}

success = true;

}

if (!success) return false;

this.add('-activate', source, 'move: Court Change');

},

secondary: null,

target: "all",

type: "Normal",

},

covet: {

num: 343,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "If this attack was successful and the user has not fainted, it steals the target's held item if the user is not holding one. The target's item is not stolen if it is a Mail or Z-Crystal, or if the target is a Kyogre holding a Blue Orb, a Groudon holding a Red Orb, a Giratina holding a Griseous Orb, an Arceus holding a Plate, a Genesect holding a Drive, a Silvally holding a Memory, or a Pokemon that can Mega Evolve holding the Mega Stone for its species. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "If the user has no item, it steals the target's.",

name: "Covet",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onAfterHit(target, source, move) {

if (source.item || source.volatiles['gem']) {

return;

}

const yourItem = target.takeItem(source);

if (!yourItem) {

return;

}

if (

!this.singleEvent('TakeItem', yourItem, target.itemData, source, target, move, yourItem) ||

!source.setItem(yourItem)

) {

target.item = yourItem.id; // bypass setItem so we don't break choicelock or anything

return;

}

this.add('-item', source, yourItem, '[from] move: Covet', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

crabhammer: {

num: 152,

accuracy: 90,

basePower: 100,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Crabhammer",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Water",

contestType: "Tough",

},

craftyshield: {

num: 578,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and its party members are protected from non-damaging attacks made by other Pokemon, including allies, during this turn. Fails if the user moves last this turn or if this move is already in effect for the user's side.",

shortDesc: "Protects allies from Status moves this turn.",

name: "Crafty Shield",

pp: 10,

priority: 3,

flags: {},

sideCondition: 'craftyshield',

onTryHitSide(side, source) {

return !!this.queue.willAct();

},

condition: {

duration: 1,

onStart(target, source) {

this.add('-singleturn', source, 'Crafty Shield');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (['self', 'all'].includes(move.target) || move.category !== 'Status') return;

this.add('-activate', target, 'move: Crafty Shield');

return this.NOT\_FAIL;

},

},

secondary: null,

target: "allySide",

type: "Fairy",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

crosschop: {

num: 238,

accuracy: 80,

basePower: 100,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Cross Chop",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

crosspoison: {

num: 440,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a 10% chance to poison the target and a higher chance for a critical hit.",

shortDesc: "High critical hit ratio. 10% chance to poison.",

name: "Cross Poison",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'psn',

},

critRatio: 2,

target: "normal",

type: "Poison",

contestType: "Cool",

},

crunch: {

num: 242,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 20% chance to lower the target's Defense by 1 stage.",

shortDesc: "20% chance to lower the target's Defense by 1.",

name: "Crunch",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

boosts: {

def: -1,

},

},

target: "normal",

type: "Dark",

contestType: "Tough",

},

crushclaw: {

num: 306,

accuracy: 95,

basePower: 75,

category: "Physical",

desc: "Has a 50% chance to lower the target's Defense by 1 stage.",

shortDesc: "50% chance to lower the target's Defense by 1.",

name: "Crush Claw",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

def: -1,

},

},

target: "normal",

type: "Normal",

contestType: "Cool",

},

crushgrip: {

num: 462,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

return Math.floor(Math.floor((120 \* (100 \* Math.floor(target.hp \* 4096 / target.maxhp)) + 2048 - 1) / 4096) / 100) || 1;

},

category: "Physical",

desc: "Power is equal to 120 \* (target's current HP / target's maximum HP), rounded half down, but not less than 1.",

shortDesc: "More power the more HP the target has left.",

isNonstandard: "Past",

name: "Crush Grip",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 190},

maxMove: {basePower: 140},

contestType: "Tough",

},

curse: {

num: 174,

accuracy: true,

basePower: 0,

category: "Status",

desc: "If the user is not a Ghost type, lowers the user's Speed by 1 stage and raises the user's Attack and Defense by 1 stage. If the user is a Ghost type, the user loses 1/2 of its maximum HP, rounded down and even if it would cause fainting, in exchange for the target losing 1/4 of its maximum HP, rounded down, at the end of each turn while it is active. If the target uses Baton Pass, the replacement will continue to be affected. Fails if there is no target or if the target is already affected.",

shortDesc: "Curses if Ghost, else -1 Spe, +1 Atk, +1 Def.",

name: "Curse",

pp: 10,

priority: 0,

flags: {authentic: 1},

volatileStatus: 'curse',

onModifyMove(move, source, target) {

if (!source.hasType('Ghost')) {

move.target = move.nonGhostTarget as MoveTarget;

}

},

onTryHit(target, source, move) {

if (!source.hasType('Ghost')) {

delete move.volatileStatus;

delete move.onHit;

move.self = {boosts: {spe: -1, atk: 1, def: 1}};

} else if (move.volatileStatus && target.volatiles.curse) {

return false;

}

},

onHit(target, source) {

this.directDamage(source.maxhp / 2, source, source);

},

condition: {

onStart(pokemon, source) {

this.add('-start', pokemon, 'Curse', '[of] ' + source);

},

onResidualOrder: 10,

onResidual(pokemon) {

this.damage(pokemon.baseMaxhp / 4);

},

},

secondary: null,

target: "randomNormal",

nonGhostTarget: "self",

type: "Ghost",

zMove: {effect: 'curse'},

contestType: "Tough",

},

cut: {

num: 15,

accuracy: 95,

basePower: 50,

category: "Physical",

shortDesc: "No additional effect.",

name: "Cut",

pp: 30,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

darkestlariat: {

num: 663,

accuracy: 100,

basePower: 85,

category: "Physical",

desc: "Ignores the target's stat stage changes, including evasiveness.",

shortDesc: "Ignores the target's stat stage changes.",

name: "Darkest Lariat",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ignoreEvasion: true,

ignoreDefensive: true,

secondary: null,

target: "normal",

type: "Dark",

contestType: "Cool",

},

darkpulse: {

num: 399,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 20% chance to flinch the target.",

shortDesc: "20% chance to flinch the target.",

name: "Dark Pulse",

pp: 15,

priority: 0,

flags: {protect: 1, pulse: 1, mirror: 1, distance: 1},

secondary: {

chance: 20,

volatileStatus: 'flinch',

},

target: "any",

type: "Dark",

contestType: "Cool",

},

darkvoid: {

num: 464,

accuracy: 50,

basePower: 0,

category: "Status",

desc: "Causes the target to fall asleep. This move cannot be used successfully unless the user's current form, while considering Transform, is Darkrai.",

shortDesc: "Darkrai: Causes the foe(s) to fall asleep.",

isNonstandard: "Past",

name: "Dark Void",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'slp',

onTryMove(pokemon, target, move) {

if (pokemon.species.name === 'Darkrai' || move.hasBounced) {

return;

}

this.add('-fail', pokemon, 'move: Dark Void');

this.hint("Only a Pokemon whose form is Darkrai can use this move.");

return null;

},

secondary: null,

target: "allAdjacentFoes",

type: "Dark",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

dazzlinggleam: {

num: 605,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent foes.",

name: "Dazzling Gleam",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Fairy",

contestType: "Beautiful",

},

decorate: {

num: 777,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the target's Attack and Special Attack by 2 stages.",

shortDesc: "Raises the target's Attack and Sp. Atk by 2.",

name: "Decorate",

pp: 15,

priority: 0,

flags: {mystery: 1},

secondary: null,

boosts: {

atk: 2,

spa: 2,

},

target: "normal",

type: "Fairy",

},

defendorder: {

num: 455,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense and Special Defense by 1 stage.",

shortDesc: "Raises the user's Defense and Sp. Def by 1.",

name: "Defend Order",

pp: 10,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 1,

spd: 1,

},

secondary: null,

target: "self",

type: "Bug",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

defensecurl: {

num: 111,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 1 stage. As long as the user remains active, the power of the user's Ice Ball and Rollout will be doubled (this effect is not stackable).",

shortDesc: "Raises the user's Defense by 1.",

name: "Defense Curl",

pp: 40,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 1,

},

volatileStatus: 'defensecurl',

condition: {

noCopy: true,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {accuracy: 1}},

contestType: "Cute",

},

defog: {

num: 432,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Lowers the target's evasiveness by 1 stage. If this move is successful and whether or not the target's evasiveness was affected, the effects of Reflect, Light Screen, Aurora Veil, Safeguard, Mist, Spikes, Toxic Spikes, Stealth Rock, and Sticky Web end for the target's side, and the effects of Spikes, Toxic Spikes, Stealth Rock, and Sticky Web end for the user's side. Ignores a target's substitute, although a substitute will still block the lowering of evasiveness. If there is a terrain active and this move is successful, the terrain will be cleared.",

shortDesc: "-1 evasion; clears terrain and hazards on both sides.",

name: "Defog",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

onHit(target, source, move) {

let success = false;

if (!target.volatiles['substitute'] || move.infiltrates) success = !!this.boost({evasion: -1});

const removeTarget = [

'reflect', 'lightscreen', 'auroraveil', 'safeguard', 'mist', 'spikes', 'toxicspikes', 'stealthrock', 'stickyweb', 'gmaxsteelsurge',

];

const removeAll = [

'spikes', 'toxicspikes', 'stealthrock', 'stickyweb', 'gmaxsteelsurge',

];

for (const targetCondition of removeTarget) {

if (target.side.removeSideCondition(targetCondition)) {

if (!removeAll.includes(targetCondition)) continue;

this.add('-sideend', target.side, this.dex.getEffect(targetCondition).name, '[from] move: Defog', '[of] ' + source);

success = true;

}

}

for (const sideCondition of removeAll) {

if (source.side.removeSideCondition(sideCondition)) {

this.add('-sideend', source.side, this.dex.getEffect(sideCondition).name, '[from] move: Defog', '[of] ' + source);

success = true;

}

}

this.field.clearTerrain();

return success;

},

secondary: null,

target: "normal",

type: "Flying",

zMove: {boost: {accuracy: 1}},

contestType: "Cool",

},

destinybond: {

num: 194,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the user's next move, if an opposing Pokemon's attack knocks the user out, that Pokemon faints as well, unless the attack was Doom Desire or Future Sight. Fails if the user used this move successfully as its last move, disregarding moves used through the Dancer Ability.",

shortDesc: "If an opponent knocks out the user, it also faints.",

name: "Destiny Bond",

pp: 5,

priority: 0,

flags: {authentic: 1},

volatileStatus: 'destinybond',

onPrepareHit(pokemon) {

return !pokemon.removeVolatile('destinybond');

},

condition: {

onStart(pokemon) {

this.add('-singlemove', pokemon, 'Destiny Bond');

},

onFaint(target, source, effect) {

if (!source || !effect || target.side === source.side) return;

if (effect.effectType === 'Move' && !effect.isFutureMove) {

if (source.volatiles['dynamax']) {

this.add('-hint', "Dynamaxed Pokémon are immune to Destiny Bond.");

return;

}

this.add('-activate', target, 'move: Destiny Bond');

source.faint();

}

},

onBeforeMovePriority: -1,

onBeforeMove(pokemon, target, move) {

if (move.id === 'destinybond') return;

this.debug('removing Destiny Bond before attack');

pokemon.removeVolatile('destinybond');

},

onMoveAborted(pokemon, target, move) {

pokemon.removeVolatile('destinybond');

},

},

secondary: null,

target: "self",

type: "Ghost",

zMove: {effect: 'redirect'},

contestType: "Clever",

},

detect: {

num: 197,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Prevents moves from affecting the user this turn.",

name: "Detect",

pp: 5,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'protect',

onPrepareHit(pokemon) {

return !!this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

secondary: null,

target: "self",

type: "Fighting",

zMove: {boost: {evasion: 1}},

contestType: "Cool",

},

devastatingdrake: {

num: 652,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Devastating Drake",

pp: 1,

priority: 0,

flags: {},

isZ: "dragoniumz",

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Cool",

},

diamondstorm: {

num: 591,

accuracy: 95,

basePower: 100,

category: "Physical",

desc: "Has a 50% chance to raise the user's Defense by 2 stages.",

shortDesc: "50% chance to raise user's Def by 2 for each hit.",

isNonstandard: "Past",

name: "Diamond Storm",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 50,

self: {

boosts: {

def: 2,

},

},

},

target: "allAdjacentFoes",

type: "Rock",

contestType: "Beautiful",

},

dig: {

num: 91,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks other than Earthquake and Magnitude but takes double damage from them, and is also unaffected by weather. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Digs underground turn 1, strikes turn 2.",

name: "Dig",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1, nonsky: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onImmunity(type, pokemon) {

if (type === 'sandstorm' || type === 'hail') return false;

},

onInvulnerability(target, source, move) {

if (['earthquake', 'magnitude'].includes(move.id)) {

return;

}

return false;

},

onSourceModifyDamage(damage, source, target, move) {

if (move.id === 'earthquake' || move.id === 'magnitude') {

return this.chainModify(2);

}

},

},

secondary: null,

target: "normal",

type: "Ground",

contestType: "Tough",

},

disable: {

num: 50,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "For 4 turns, the target's last move used becomes disabled. Fails if one of the target's moves is already disabled, if the target has not made a move, if the target no longer knows the move, or if the move was a Max or G-Max Move.",

shortDesc: "For 4 turns, disables the target's last move used.",

name: "Disable",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'disable',

onTryHit(target) {

if (!target.lastMove || target.lastMove.isZ || target.lastMove.isMax) {

return false;

}

},

condition: {

duration: 5,

noCopy: true, // doesn't get copied by Baton Pass

onStart(pokemon, source, effect) {

// The target hasn't taken its turn, or Cursed Body activated and the move was not used through Dancer or Instruct

if (

this.queue.willMove(pokemon) ||

(pokemon === this.activePokemon && this.activeMove && !this.activeMove.isExternal)

) {

this.effectData.duration--;

}

if (!pokemon.lastMove) {

this.debug('pokemon hasn\'t moved yet');

return false;

}

for (const moveSlot of pokemon.moveSlots) {

if (moveSlot.id === pokemon.lastMove.id) {

if (!moveSlot.pp) {

this.debug('Move out of PP');

return false;

} else {

if (effect.id === 'cursedbody') {

this.add('-start', pokemon, 'Disable', moveSlot.move, '[from] ability: Cursed Body', '[of] ' + source);

} else {

this.add('-start', pokemon, 'Disable', moveSlot.move);

}

this.effectData.move = pokemon.lastMove.id;

return;

}

}

}

// this can happen if Disable works on a Z-move

return false;

},

onResidualOrder: 14,

onEnd(pokemon) {

this.add('-end', pokemon, 'Disable');

},

onBeforeMovePriority: 7,

onBeforeMove(attacker, defender, move) {

if (!move.isZ && move.id === this.effectData.move) {

this.add('cant', attacker, 'Disable', move);

return false;

}

},

onDisableMove(pokemon) {

for (const moveSlot of pokemon.moveSlots) {

if (moveSlot.id === this.effectData.move) {

pokemon.disableMove(moveSlot.id);

}

}

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

disarmingvoice: {

num: 574,

accuracy: true,

basePower: 40,

category: "Special",

desc: "This move does not check accuracy.",

shortDesc: "This move does not check accuracy. Hits foes.",

name: "Disarming Voice",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Fairy",

contestType: "Cute",

},

discharge: {

num: 435,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze adjacent Pokemon.",

name: "Discharge",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "allAdjacent",

type: "Electric",

contestType: "Beautiful",

},

dive: {

num: 291,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks other than Surf and Whirlpool but takes double damage from them, and is also unaffected by weather. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Dives underwater turn 1, strikes turn 2.",

name: "Dive",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1, nonsky: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

if (attacker.hasAbility('gulpmissile') && attacker.species.name === 'Cramorant' && !attacker.transformed) {

const forme = attacker.hp <= attacker.maxhp / 2 ? 'cramorantgorging' : 'cramorantgulping';

attacker.formeChange(forme, move);

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onImmunity(type, pokemon) {

if (type === 'sandstorm' || type === 'hail') return false;

},

onInvulnerability(target, source, move) {

if (['surf', 'whirlpool'].includes(move.id)) {

return;

}

return false;

},

onSourceModifyDamage(damage, source, target, move) {

if (move.id === 'surf' || move.id === 'whirlpool') {

return this.chainModify(2);

}

},

},

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

dizzypunch: {

num: 146,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a 20% chance to confuse the target.",

shortDesc: "20% chance to confuse the target.",

isNonstandard: "Past",

name: "Dizzy Punch",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 20,

volatileStatus: 'confusion',

},

target: "normal",

type: "Normal",

contestType: "Cute",

},

doomdesire: {

num: 353,

accuracy: 100,

basePower: 140,

category: "Special",

desc: "Deals damage two turns after this move is used. At the end of that turn, the damage is calculated at that time and dealt to the Pokemon at the position the target had when the move was used. If the user is no longer active at the time, damage is calculated based on the user's natural Special Attack stat, types, and level, with no boosts from its held item or Ability. Fails if this move or Future Sight is already in effect for the target's position.",

shortDesc: "Hits two turns after being used.",

name: "Doom Desire",

pp: 5,

priority: 0,

flags: {},

isFutureMove: true,

onTry(source, target) {

if (!target.side.addSlotCondition(target, 'futuremove')) return false;

Object.assign(target.side.slotConditions[target.position]['futuremove'], {

move: 'doomdesire',

source: source,

moveData: {

id: 'doomdesire',

name: "Doom Desire",

accuracy: 100,

basePower: 140,

category: "Special",

priority: 0,

flags: {},

effectType: 'Move',

isFutureMove: true,

type: 'Steel',

},

});

this.add('-start', source, 'Doom Desire');

return null;

},

secondary: null,

target: "normal",

type: "Steel",

contestType: "Beautiful",

},

doubleedge: {

num: 38,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 33% the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 33% recoil.",

name: "Double-Edge",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [33, 100],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

doublehit: {

num: 458,

accuracy: 90,

basePower: 35,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Double Hit",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 140},

maxMove: {basePower: 120},

contestType: "Cool",

},

doubleironbash: {

num: 742,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit. Has a 30% chance to flinch the target.",

shortDesc: "Hits twice. 30% chance to flinch.",

name: "Double Iron Bash",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

multihit: 2,

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Steel",

zMove: {basePower: 180},

maxMove: {basePower: 140},

contestType: "Clever",

},

doublekick: {

num: 24,

accuracy: 100,

basePower: 30,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Double Kick",

pp: 30,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Fighting",

maxMove: {basePower: 80},

contestType: "Cool",

},

doubleslap: {

num: 3,

accuracy: 85,

basePower: 15,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

isNonstandard: "Past",

name: "Double Slap",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

doubleteam: {

num: 104,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's evasiveness by 1 stage.",

shortDesc: "Raises the user's evasiveness by 1.",

name: "Double Team",

pp: 15,

priority: 0,

flags: {snatch: 1},

boosts: {

evasion: 1,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cool",

},

dracometeor: {

num: 434,

accuracy: 90,

basePower: 130,

category: "Special",

desc: "Lowers the user's Special Attack by 2 stages.",

shortDesc: "Lowers the user's Sp. Atk by 2.",

name: "Draco Meteor",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

self: {

boosts: {

spa: -2,

},

},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Beautiful",

},

dragonascent: {

num: 620,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "Lowers the user's Defense and Special Defense by 1 stage.",

shortDesc: "Lowers the user's Defense and Sp. Def by 1.",

isNonstandard: "Past",

name: "Dragon Ascent",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

self: {

boosts: {

def: -1,

spd: -1,

},

},

target: "any",

type: "Flying",

contestType: "Beautiful",

},

dragonbreath: {

num: 225,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze the target.",

name: "Dragon Breath",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Dragon",

contestType: "Cool",

},

dragonclaw: {

num: 337,

accuracy: 100,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Dragon Claw",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Cool",

},

dragondance: {

num: 349,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack and Speed by 1 stage.",

shortDesc: "Raises the user's Attack and Speed by 1.",

name: "Dragon Dance",

pp: 20,

priority: 0,

flags: {snatch: 1, dance: 1},

boosts: {

atk: 1,

spe: 1,

},

secondary: null,

target: "self",

type: "Dragon",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cool",

},

dragondarts: {

num: 751,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit. In Double Battles, this move attempts to hit the targeted Pokemon and its ally once each. If hitting one of these Pokemon would be prevented by immunity, protection, semi-invulnerability, an Ability, or accuracy, it attempts to hit the other Pokemon twice instead. If this move is redirected, it hits that target twice.",

shortDesc: "Hits twice. Doubles: Tries to hit each foe once.",

name: "Dragon Darts",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: 2,

smartTarget: true,

secondary: null,

target: "normal",

type: "Dragon",

maxMove: {basePower: 130},

},

dragonhammer: {

num: 692,

accuracy: 100,

basePower: 90,

category: "Physical",

shortDesc: "No additional effect.",

name: "Dragon Hammer",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Tough",

},

dragonpulse: {

num: 406,

accuracy: 100,

basePower: 85,

category: "Special",

shortDesc: "No additional effect.",

name: "Dragon Pulse",

pp: 10,

priority: 0,

flags: {protect: 1, pulse: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Dragon",

contestType: "Beautiful",

},

dragonrage: {

num: 82,

accuracy: 100,

basePower: 0,

damage: 40,

category: "Special",

shortDesc: "Deals 40 HP of damage to the target.",

isNonstandard: "Past",

name: "Dragon Rage",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Cool",

},

dragonrush: {

num: 407,

accuracy: 75,

basePower: 100,

category: "Physical",

desc: "Has a 20% chance to flinch the target. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "20% chance to flinch the target.",

name: "Dragon Rush",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'flinch',

},

target: "normal",

type: "Dragon",

contestType: "Tough",

},

dragontail: {

num: 525,

accuracy: 90,

basePower: 60,

category: "Physical",

desc: "If both the user and the target have not fainted, the target is forced to switch out and be replaced with a random unfainted ally. This effect fails if the target used Ingrain previously, has the Suction Cups Ability, or this move hit a substitute.",

shortDesc: "Forces the target to switch to a random ally.",

name: "Dragon Tail",

pp: 10,

priority: -6,

flags: {contact: 1, protect: 1, mirror: 1},

forceSwitch: true,

target: "normal",

type: "Dragon",

contestType: "Tough",

},

drainingkiss: {

num: 577,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "The user recovers 3/4 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 75% of the damage dealt.",

name: "Draining Kiss",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, heal: 1},

drain: [3, 4],

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Cute",

},

drainpunch: {

num: 409,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Drain Punch",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

dreameater: {

num: 138,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "The target is unaffected by this move unless it is asleep. The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User gains 1/2 HP inflicted. Sleeping target only.",

name: "Dream Eater",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

onTryImmunity(target) {

return target.status === 'slp' || target.hasAbility('comatose');

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

drillpeck: {

num: 65,

accuracy: 100,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Drill Peck",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

drillrun: {

num: 529,

accuracy: 95,

basePower: 80,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Drill Run",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Ground",

contestType: "Tough",

},

drumbeating: {

num: 778,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the target's Speed by 1.",

name: "Drum Beating",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Grass",

},

dualchop: {

num: 530,

accuracy: 90,

basePower: 40,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Dual Chop",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Dragon",

maxMove: {basePower: 130},

contestType: "Tough",

},

dualwingbeat: {

num: 814,

accuracy: 90,

basePower: 40,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Dual Wingbeat",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Flying",

maxMove: {basePower: 130},

},

dynamaxcannon: {

num: 744,

accuracy: 100,

basePower: 100,

category: "Special",

shortDesc: "Damage doubles if the target is Dynamaxed.",

name: "Dynamax Cannon",

pp: 5,

priority: 0,

flags: {protect: 1},

secondary: null,

target: "normal",

type: "Dragon",

},

dynamicpunch: {

num: 223,

accuracy: 50,

basePower: 100,

category: "Physical",

desc: "Has a 100% chance to confuse the target.",

shortDesc: "100% chance to confuse the target.",

name: "Dynamic Punch",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 100,

volatileStatus: 'confusion',

},

target: "normal",

type: "Fighting",

contestType: "Cool",

},

earthpower: {

num: 414,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Earth Power",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Ground",

contestType: "Beautiful",

},

earthquake: {

num: 89,

accuracy: 100,

basePower: 100,

category: "Physical",

desc: "Damage doubles if the target is using Dig.",

shortDesc: "Hits adjacent Pokemon. Double damage on Dig.",

name: "Earthquake",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: null,

target: "allAdjacent",

type: "Ground",

contestType: "Tough",

},

echoedvoice: {

num: 497,

accuracy: 100,

basePower: 40,

basePowerCallback() {

if (this.field.pseudoWeather.echoedvoice) {

return 40 \* this.field.pseudoWeather.echoedvoice.multiplier;

}

return 40;

},

category: "Special",

desc: "For every consecutive turn that this move is used by at least one Pokemon, this move's power is multiplied by the number of turns to pass, but not more than 5.",

shortDesc: "Power increases when used on consecutive turns.",

name: "Echoed Voice",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

onTry() {

this.field.addPseudoWeather('echoedvoice');

},

condition: {

duration: 2,

onStart() {

this.effectData.multiplier = 1;

},

onRestart() {

if (this.effectData.duration !== 2) {

this.effectData.duration = 2;

if (this.effectData.multiplier < 5) {

this.effectData.multiplier++;

}

}

},

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

eerieimpulse: {

num: 598,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Special Attack by 2 stages.",

shortDesc: "Lowers the target's Sp. Atk by 2.",

name: "Eerie Impulse",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

spa: -2,

},

secondary: null,

target: "normal",

type: "Electric",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

eggbomb: {

num: 121,

accuracy: 75,

basePower: 100,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Egg Bomb",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

electricterrain: {

num: 604,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the terrain becomes Electric Terrain. During the effect, the power of Electric-type attacks made by grounded Pokemon is multiplied by 1.3 and grounded Pokemon cannot fall asleep; Pokemon already asleep do not wake up. Camouflage transforms the user into an Electric type, Nature Power becomes Thunderbolt, and Secret Power has a 30% chance to cause paralysis. Fails if the current terrain is Electric Terrain.",

shortDesc: "5 turns. Grounded: +Electric power, can't sleep.",

name: "Electric Terrain",

pp: 10,

priority: 0,

flags: {nonsky: 1},

terrain: 'electricterrain',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasItem('terrainextender')) {

return 8;

}

return 5;

},

onSetStatus(status, target, source, effect) {

if (status.id === 'slp' && target.isGrounded() && !target.isSemiInvulnerable()) {

if (effect.id === 'yawn' || (effect.effectType === 'Move' && !effect.secondaries)) {

this.add('-activate', target, 'move: Electric Terrain');

}

return false;

}

},

onTryAddVolatile(status, target) {

if (!target.isGrounded() || target.isSemiInvulnerable()) return;

if (status.id === 'yawn') {

this.add('-activate', target, 'move: Electric Terrain');

return null;

}

},

onBasePowerPriority: 6,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Electric' && attacker.isGrounded() && !attacker.isSemiInvulnerable()) {

this.debug('electric terrain boost');

return this.chainModify([0x14CD, 0x1000]);

}

},

onStart(battle, source, effect) {

if (effect?.effectType === 'Ability') {

this.add('-fieldstart', 'move: Electric Terrain', '[from] ability: ' + effect, '[of] ' + source);

} else {

this.add('-fieldstart', 'move: Electric Terrain');

}

},

onResidualOrder: 21,

onResidualSubOrder: 2,

onEnd() {

this.add('-fieldend', 'move: Electric Terrain');

},

},

secondary: null,

target: "all",

type: "Electric",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

electrify: {

num: 582,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Causes the target's move to become Electric type this turn. Among effects that can change a move's type, this effect happens last. Fails if the target already moved this turn.",

shortDesc: "Changes the target's move to Electric this turn.",

name: "Electrify",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

volatileStatus: 'electrify',

onTryHit(target) {

if (!this.queue.willMove(target) && target.activeTurns) return false;

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'move: Electrify');

},

onModifyTypePriority: -2,

onModifyType(move) {

if (move.id !== 'struggle') {

this.debug('Electrify making move type electric');

move.type = 'Electric';

}

},

},

secondary: null,

target: "normal",

type: "Electric",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

electroball: {

num: 486,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

let ratio = Math.floor(pokemon.getStat('spe') / target.getStat('spe'));

if (!isFinite(ratio)) ratio = 0;

const bp = [40, 60, 80, 120, 150][Math.min(ratio, 4)];

this.debug(`${bp} bp`);

return bp;

},

category: "Special",

desc: "The power of this move depends on (user's current Speed / target's current Speed), rounded down. Power is equal to 150 if the result is 4 or more, 120 if 3, 80 if 2, 60 if 1, 40 if less than 1. If the target's current Speed is 0, this move's power is 40.",

shortDesc: "More power the faster the user is than the target.",

name: "Electro Ball",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Electric",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cool",

},

electroweb: {

num: 527,

accuracy: 95,

basePower: 55,

category: "Special",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Speed by 1.",

name: "Electroweb",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "allAdjacentFoes",

type: "Electric",

contestType: "Beautiful",

},

embargo: {

num: 373,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "For 5 turns, the target's held item has no effect. An item's effect of causing forme changes is unaffected, but any other effects from such items are negated. During the effect, Fling and Natural Gift are prevented from being used by the target. Items thrown at the target with Fling will still activate for it. If the target uses Baton Pass, the replacement will remain unable to use items.",

shortDesc: "For 5 turns, the target's item has no effect.",

isNonstandard: "Past",

name: "Embargo",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'embargo',

condition: {

duration: 5,

onStart(pokemon) {

this.add('-start', pokemon, 'Embargo');

},

// Item suppression implemented in Pokemon.ignoringItem() within sim/pokemon.js

onResidualOrder: 18,

onEnd(pokemon) {

this.add('-end', pokemon, 'Embargo');

},

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

ember: {

num: 52,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target.",

name: "Ember",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Cute",

},

encore: {

num: 227,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "For its next 3 turns, the target is forced to repeat its last move used. If the affected move runs out of PP, the effect ends. Fails if the target is already under this effect, if it has not made a move, if the move has 0 PP, if the move is Assist, Copycat, Encore, Me First, Metronome, Mimic, Mirror Move, Nature Power, Sketch, Sleep Talk, Struggle, or Transform, or if the target is Dynamaxed.",

shortDesc: "Target repeats its last move for its next 3 turns.",

name: "Encore",

pp: 5,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'encore',

condition: {

duration: 3,

noCopy: true, // doesn't get copied by Z-Baton Pass

onStart(target) {

const noEncore = [

'assist', 'copycat', 'encore', 'mefirst', 'metronome', 'mimic', 'mirrormove', 'naturepower', 'sketch', 'sleeptalk', 'struggle', 'transform',

];

let move: Move | ActiveMove | null = target.lastMove;

if (!move || target.volatiles['dynamax']) return false;

if (move.isMax && move.baseMove) move = this.dex.getMove(move.baseMove);

const moveIndex = target.moves.indexOf(move.id);

if (move.isZ || noEncore.includes(move.id) || !target.moveSlots[moveIndex] || target.moveSlots[moveIndex].pp <= 0) {

// it failed

return false;

}

this.effectData.move = move.id;

this.add('-start', target, 'Encore');

if (!this.queue.willMove(target)) {

this.effectData.duration++;

}

},

onOverrideAction(pokemon, target, move) {

if (move.id !== this.effectData.move) return this.effectData.move;

},

onResidualOrder: 13,

onResidual(target) {

if (target.moves.includes(this.effectData.move) &&

target.moveSlots[target.moves.indexOf(this.effectData.move)].pp <= 0) {

// early termination if you run out of PP

target.removeVolatile('encore');

}

},

onEnd(target) {

this.add('-end', target, 'Encore');

},

onDisableMove(pokemon) {

if (!this.effectData.move || !pokemon.hasMove(this.effectData.move)) {

return;

}

for (const moveSlot of pokemon.moveSlots) {

if (moveSlot.id !== this.effectData.move) {

pokemon.disableMove(moveSlot.id);

}

}

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Cute",

},

endeavor: {

num: 283,

accuracy: 100,

basePower: 0,

damageCallback(pokemon, target) {

return target.getUndynamaxedHP() - pokemon.hp;

},

category: "Physical",

desc: "Deals damage to the target equal to (target's current HP - user's current HP). The target is unaffected if its current HP is less than or equal to the user's current HP.",

shortDesc: "Lowers the target's HP to the user's HP.",

name: "Endeavor",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onTryImmunity(target, pokemon) {

return pokemon.hp < target.hp;

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Tough",

},

endure: {

num: 203,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user will survive attacks made by other Pokemon during this turn with at least 1 HP. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "User survives attacks this turn with at least 1 HP.",

name: "Endure",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'endure',

onTryHit(pokemon) {

return this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'move: Endure');

},

onDamagePriority: -10,

onDamage(damage, target, source, effect) {

if (effect?.effectType === 'Move' && damage >= target.hp) {

this.add('-activate', target, 'move: Endure');

return target.hp - 1;

}

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

energyball: {

num: 412,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Energy Ball",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Grass",

contestType: "Beautiful",

},

entrainment: {

num: 494,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target's Ability to become the same as the user's. Fails if the target's Ability is Battle Bond, Comatose, Disguise, Multitype, Power Construct, RKS System, Schooling, Shields Down, Stance Change, Truant, or the same Ability as the user, or if the user's Ability is Battle Bond, Comatose, Disguise, Flower Gift, Forecast, Illusion, Imposter, Multitype, Neutralizing Gas, Power Construct, Power of Alchemy, Receiver, RKS System, Schooling, Shields Down, Stance Change, Trace, or Zen Mode.",

shortDesc: "The target's Ability changes to match the user's.",

name: "Entrainment",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onTryHit(target, source) {

if (target === source || target.volatiles['dynamax']) return false;

const bannedTargetAbilities = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'truant',

];

const bannedSourceAbilities = [

'battlebond', 'comatose', 'disguise', 'flowergift', 'forecast', 'illusion', 'imposter', 'multitype', 'neutralizinggas', 'powerconstruct', 'powerofalchemy', 'receiver', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'trace', 'zenmode',

];

if (

bannedTargetAbilities.includes(target.ability) || bannedSourceAbilities.includes(source.ability) ||

target.ability === source.ability

) {

return false;

}

},

onHit(target, source) {

const oldAbility = target.setAbility(source.ability);

if (oldAbility) {

this.add('-ability', target, target.getAbility().name, '[from] move: Entrainment');

if (target.side !== source.side) target.staleness = 'external';

return;

}

return false;

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

eruption: {

num: 284,

accuracy: 100,

basePower: 150,

basePowerCallback(pokemon, target, move) {

return move.basePower \* pokemon.hp / pokemon.maxhp;

},

category: "Special",

desc: "Power is equal to (user's current HP \* 150 / user's maximum HP), rounded down, but not less than 1.",

shortDesc: "Less power as user's HP decreases. Hits foe(s).",

name: "Eruption",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Fire",

contestType: "Beautiful",

},

eternabeam: {

num: 795,

accuracy: 90,

basePower: 160,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Eternabeam",

pp: 10,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Dragon",

},

expandingforce: {

num: 797,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "If the current terrain is Psychic Terrain and the user is grounded, this move hits all opposing Pokemon and has its power multiplied by 1.5.",

shortDesc: "User on Psychic Terrain: 1.5x power, hits foes.",

name: "Expanding Force",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower, source) {

if (this.field.isTerrain('psychicterrain') && source.isGrounded()) {

this.debug('terrain buff');

return this.chainModify(1.5);

}

},

onModifyMove(move, source, target) {

if (this.field.isTerrain('psychicterrain') && source.isGrounded()) {

move.target = 'allAdjacentFoes';

}

},

secondary: null,

target: "normal",

type: "Psychic",

},

explosion: {

num: 153,

accuracy: 100,

basePower: 250,

category: "Physical",

desc: "The user faints after using this move, even if this move fails for having no target. This move is prevented from executing if any active Pokemon has the Damp Ability.",

shortDesc: "Hits adjacent Pokemon. The user faints.",

name: "Explosion",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

selfdestruct: "always",

secondary: null,

target: "allAdjacent",

type: "Normal",

contestType: "Beautiful",

},

extrasensory: {

num: 326,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 10% chance to flinch the target.",

shortDesc: "10% chance to flinch the target.",

name: "Extrasensory",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'flinch',

},

target: "normal",

type: "Psychic",

contestType: "Cool",

},

extremeevoboost: {

num: 702,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack, Defense, Special Attack, Special Defense, and Speed by 2 stages.",

shortDesc: "Raises user's Atk, Def, SpA, SpD, and Spe by 2.",

isNonstandard: "Past",

name: "Extreme Evoboost",

pp: 1,

priority: 0,

flags: {},

isZ: "eeviumz",

boosts: {

atk: 2,

def: 2,

spa: 2,

spd: 2,

spe: 2,

},

secondary: null,

target: "self",

type: "Normal",

contestType: "Beautiful",

},

extremespeed: {

num: 245,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Nearly always goes first.",

name: "Extreme Speed",

pp: 5,

priority: 2,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

facade: {

num: 263,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Power doubles if the user is burned, paralyzed, or poisoned. The physical damage halving effect from the user's burn is ignored.",

shortDesc: "Power doubles if user is burn/poison/paralyzed.",

name: "Facade",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onBasePower(basePower, pokemon) {

if (pokemon.status && pokemon.status !== 'slp') {

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

fairylock: {

num: 587,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Prevents all active Pokemon from switching next turn. A Pokemon can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. Fails if the effect is already active.",

shortDesc: "Prevents all Pokemon from switching next turn.",

name: "Fairy Lock",

pp: 10,

priority: 0,

flags: {mirror: 1, authentic: 1},

pseudoWeather: 'fairylock',

condition: {

duration: 2,

onStart(target) {

this.add('-fieldactivate', 'move: Fairy Lock');

},

onTrapPokemon(pokemon) {

pokemon.tryTrap();

},

},

secondary: null,

target: "all",

type: "Fairy",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

fairywind: {

num: 584,

accuracy: 100,

basePower: 40,

category: "Special",

shortDesc: "No additional effect.",

name: "Fairy Wind",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Beautiful",

},

fakeout: {

num: 252,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "Has a 100% chance to flinch the target. Fails unless it is the user's first turn on the field.",

shortDesc: "Hits first. First turn out only. 100% flinch chance.",

name: "Fake Out",

pp: 10,

priority: 3,

flags: {contact: 1, protect: 1, mirror: 1},

onTry(pokemon, target) {

if (pokemon.activeMoveActions > 1) {

this.attrLastMove('[still]');

this.add('-fail', pokemon);

this.hint("Fake Out only works on your first turn out.");

return null;

}

},

secondary: {

chance: 100,

volatileStatus: 'flinch',

},

target: "normal",

type: "Normal",

contestType: "Cute",

},

faketears: {

num: 313,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Special Defense by 2 stages.",

shortDesc: "Lowers the target's Sp. Def by 2.",

name: "Fake Tears",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

boosts: {

spd: -2,

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {spa: 1}},

contestType: "Cute",

},

falsesurrender: {

num: 793,

accuracy: true,

basePower: 80,

category: "Physical",

shortDesc: "This move does not check accuracy.",

name: "False Surrender",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

},

falseswipe: {

num: 206,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "Leaves the target with at least 1 HP.",

shortDesc: "Always leaves the target with at least 1 HP.",

name: "False Swipe",

pp: 40,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

noFaint: true,

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

featherdance: {

num: 297,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 2 stages.",

shortDesc: "Lowers the target's Attack by 2.",

name: "Feather Dance",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1, dance: 1},

boosts: {

atk: -2,

},

secondary: null,

target: "normal",

type: "Flying",

zMove: {boost: {def: 1}},

contestType: "Beautiful",

},

feint: {

num: 364,

accuracy: 100,

basePower: 30,

category: "Physical",

desc: "If this move is successful, it breaks through the target's Baneful Bunker, Detect, King's Shield, Protect, or Spiky Shield for this turn, allowing other Pokemon to attack the target normally. If the target's side is protected by Crafty Shield, Mat Block, Quick Guard, or Wide Guard, that protection is also broken for this turn and other Pokemon may attack the target's side normally.",

shortDesc: "Nullifies Detect, Protect, and Quick/Wide Guard.",

name: "Feint",

pp: 10,

priority: 2,

flags: {mirror: 1},

breaksProtect: true,

// Breaking protection implemented in scripts.js

secondary: null,

target: "normal",

type: "Normal",

contestType: "Clever",

},

feintattack: {

num: 185,

accuracy: true,

basePower: 60,

category: "Physical",

shortDesc: "This move does not check accuracy.",

isNonstandard: "Past",

name: "Feint Attack",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

fellstinger: {

num: 565,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Raises the user's Attack by 3 stages if this move knocks out the target.",

shortDesc: "Raises user's Attack by 3 if this KOes the target.",

name: "Fell Stinger",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onAfterMoveSecondarySelf(pokemon, target, move) {

if (!target || target.fainted || target.hp <= 0) this.boost({atk: 3}, pokemon, pokemon, move);

},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cool",

},

fierydance: {

num: 552,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 50% chance to raise the user's Special Attack by 1 stage.",

shortDesc: "50% chance to raise the user's Sp. Atk by 1.",

name: "Fiery Dance",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, dance: 1},

secondary: {

chance: 50,

self: {

boosts: {

spa: 1,

},

},

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

finalgambit: {

num: 515,

accuracy: 100,

basePower: 0,

damageCallback(pokemon) {

const damage = pokemon.hp;

pokemon.faint();

return damage;

},

category: "Special",

desc: "Deals damage to the target equal to the user's current HP. If this move is successful, the user faints.",

shortDesc: "Does damage equal to the user's HP. User faints.",

name: "Final Gambit",

pp: 5,

priority: 0,

flags: {protect: 1},

selfdestruct: "ifHit",

secondary: null,

target: "normal",

type: "Fighting",

zMove: {basePower: 180},

contestType: "Tough",

},

fireblast: {

num: 126,

accuracy: 85,

basePower: 110,

category: "Special",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target.",

name: "Fire Blast",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

firefang: {

num: 424,

accuracy: 95,

basePower: 65,

category: "Physical",

desc: "Has a 10% chance to burn the target and a 10% chance to flinch it.",

shortDesc: "10% chance to burn. 10% chance to flinch.",

name: "Fire Fang",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondaries: [

{

chance: 10,

status: 'brn',

}, {

chance: 10,

volatileStatus: 'flinch',

},

],

target: "normal",

type: "Fire",

contestType: "Cool",

},

firelash: {

num: 680,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 100% chance to lower the target's Defense by 1 stage.",

shortDesc: "100% chance to lower the target's Defense by 1.",

name: "Fire Lash",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

def: -1,

},

},

target: "normal",

type: "Fire",

contestType: "Cute",

},

firepledge: {

num: 519,

accuracy: 100,

basePower: 80,

basePowerCallback(target, source, move) {

if (['grasspledge', 'waterpledge'].includes(move.sourceEffect)) {

this.add('-combine');

return 150;

}

return 80;

},

category: "Special",

desc: "If one of the user's allies chose to use Grass Pledge or Water Pledge this turn and has not moved yet, it takes its turn immediately after the user and the user's move does nothing. If combined with Grass Pledge, the ally uses Fire Pledge with 150 power and a sea of fire appears on the target's side for 4 turns, which causes damage to non-Fire types equal to 1/8 of their maximum HP, rounded down, at the end of each turn during effect, including the last turn. If combined with Water Pledge, the ally uses Water Pledge with 150 power and a rainbow appears on the user's side for 4 turns, which doubles secondary effect chances but does not stack with the Serene Grace Ability. When used as a combined move, this move gains STAB no matter what the user's type is. This move does not consume the user's Fire Gem.",

shortDesc: "Use with Grass or Water Pledge for added effect.",

name: "Fire Pledge",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onPrepareHit(target, source, move) {

for (const action of this.queue) {

if (

// @ts-ignore

!action.move || !action.pokemon || !action.pokemon.isActive ||

// @ts-ignore

action.pokemon.fainted || action.maxMove || action.zmove

) {

continue;

}

// @ts-ignore

if (action.pokemon.side === source.side && ['grasspledge', 'waterpledge'].includes(action.move.id)) {

// @ts-ignore

this.queue.prioritizeAction(action, move);

this.add('-waiting', source, action.pokemon);

return null;

}

}

},

onModifyMove(move) {

if (move.sourceEffect === 'waterpledge') {

move.type = 'Water';

move.forceSTAB = true;

move.self = {sideCondition: 'waterpledge'};

}

if (move.sourceEffect === 'grasspledge') {

move.type = 'Fire';

move.forceSTAB = true;

move.sideCondition = 'firepledge';

}

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'Fire Pledge');

},

onEnd(targetSide) {

for (const pokemon of targetSide.active) {

if (pokemon && !pokemon.hasType('Fire')) {

this.damage(pokemon.baseMaxhp / 8, pokemon);

}

}

this.add('-sideend', targetSide, 'Fire Pledge');

},

onResidualOrder: 5,

onResidualSubOrder: 1,

onResidual(side) {

for (const pokemon of side.active) {

if (pokemon && !pokemon.hasType('Fire')) {

this.damage(pokemon.baseMaxhp / 8, pokemon);

}

}

},

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

firepunch: {

num: 7,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target.",

name: "Fire Punch",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Tough",

},

firespin: {

num: 83,

accuracy: 85,

basePower: 35,

category: "Special",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Fire Spin",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

firstimpression: {

num: 660,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "Fails unless it is the user's first turn on the field.",

shortDesc: "Hits first. First turn out only.",

name: "First Impression",

pp: 10,

priority: 2,

flags: {contact: 1, protect: 1, mirror: 1},

onTry(pokemon, target) {

if (pokemon.activeMoveActions > 1) {

this.add('-fail', pokemon);

this.attrLastMove('[still]');

this.hint("First Impression only works on your first turn out.");

return null;

}

},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cute",

},

fishiousrend: {

num: 755,

accuracy: 100,

basePower: 85,

basePowerCallback(pokemon, target, move) {

if (target.newlySwitched || this.queue.willMove(target)) {

this.debug('Fishious Rend damage boost');

return move.basePower \* 2;

}

this.debug('Fishious Rend NOT boosted');

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the user moves before the target.",

shortDesc: "Power doubles if user moves before the target.",

name: "Fishious Rend",

pp: 10,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

},

fissure: {

num: 90,

accuracy: 30,

basePower: 0,

category: "Physical",

desc: "Deals damage to the target equal to the target's maximum HP. Ignores accuracy and evasiveness modifiers. This attack's accuracy is equal to (user's level - target's level + 30)%, and fails if the target is at a higher level. Pokemon with the Sturdy Ability are immune.",

shortDesc: "OHKOs the target. Fails if user is a lower level.",

name: "Fissure",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

ohko: true,

secondary: null,

target: "normal",

type: "Ground",

zMove: {basePower: 180},

maxMove: {basePower: 130},

contestType: "Tough",

},

flail: {

num: 175,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const ratio = pokemon.hp \* 48 / pokemon.maxhp;

if (ratio < 2) {

return 200;

}

if (ratio < 5) {

return 150;

}

if (ratio < 10) {

return 100;

}

if (ratio < 17) {

return 80;

}

if (ratio < 33) {

return 40;

}

return 20;

},

category: "Physical",

desc: "The power of this move is 20 if X is 33 to 48, 40 if X is 17 to 32, 80 if X is 10 to 16, 100 if X is 5 to 9, 150 if X is 2 to 4, and 200 if X is 0 or 1, where X is equal to (user's current HP \* 48 / user's maximum HP), rounded down.",

shortDesc: "More power the less HP the user has left.",

name: "Flail",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cute",

},

flameburst: {

num: 481,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "If this move is successful, the target's ally loses 1/16 of its maximum HP, rounded down, unless it has the Magic Guard Ability.",

shortDesc: "Damages Pokemon next to the target as well.",

isNonstandard: "Past",

name: "Flame Burst",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit(target, source, move) {

if (target.side.active.length === 1) {

return;

}

for (const ally of target.side.active) {

if (ally && this.isAdjacent(target, ally)) {

this.damage(ally.baseMaxhp / 16, ally, source, this.dex.getEffect('Flame Burst'));

}

}

},

onAfterSubDamage(damage, target, source, move) {

if (target.side.active.length === 1) {

return;

}

for (const ally of target.side.active) {

if (ally && this.isAdjacent(target, ally)) {

this.damage(ally.baseMaxhp / 16, ally, source, this.dex.getEffect('Flame Burst'));

}

}

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

flamecharge: {

num: 488,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Has a 100% chance to raise the user's Speed by 1 stage.",

shortDesc: "100% chance to raise the user's Speed by 1.",

name: "Flame Charge",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

self: {

boosts: {

spe: 1,

},

},

},

target: "normal",

type: "Fire",

contestType: "Cool",

},

flamewheel: {

num: 172,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target. Thaws user.",

name: "Flame Wheel",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, defrost: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

flamethrower: {

num: 53,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target.",

name: "Flamethrower",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

flareblitz: {

num: 394,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "Has a 10% chance to burn the target. If the target lost HP, the user takes recoil damage equal to 33% the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 33% recoil. 10% chance to burn. Thaws user.",

name: "Flare Blitz",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, defrost: 1},

recoil: [33, 100],

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Cool",

},

flash: {

num: 148,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's accuracy by 1 stage.",

shortDesc: "Lowers the target's accuracy by 1.",

isNonstandard: "Past",

name: "Flash",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

accuracy: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {evasion: 1}},

contestType: "Beautiful",

},

flashcannon: {

num: 430,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Flash Cannon",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Steel",

contestType: "Beautiful",

},

flatter: {

num: 260,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Raises the target's Special Attack by 1 stage and confuses it.",

shortDesc: "Raises the target's Sp. Atk by 1 and confuses it.",

name: "Flatter",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

volatileStatus: 'confusion',

boosts: {

spa: 1,

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

fleurcannon: {

num: 705,

accuracy: 90,

basePower: 130,

category: "Special",

desc: "Lowers the user's Special Attack by 2 stages.",

shortDesc: "Lowers the user's Sp. Atk by 2.",

name: "Fleur Cannon",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

self: {

boosts: {

spa: -2,

},

},

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Beautiful",

},

fling: {

num: 374,

accuracy: 100,

basePower: 0,

category: "Physical",

desc: "The power of this move is based on the user's held item. The held item is lost and it activates for the target if applicable. If there is no target or the target avoids this move by protecting itself, the user's held item is still lost. The user can regain a thrown item with Recycle or the Harvest Ability. Fails if the user has no held item, if the held item cannot be thrown, if the user is under the effect of Embargo or Magic Room, or if the user has the Klutz Ability.",

shortDesc: "Flings the user's item at the target. Power varies.",

name: "Fling",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

onPrepareHit(target, source, move) {

if (source.ignoringItem()) return false;

const item = source.getItem();

if (!this.singleEvent('TakeItem', item, source.itemData, source, source, move, item)) return false;

if (!item.fling) return false;

move.basePower = item.fling.basePower;

if (item.isBerry) {

move.onHit = function (foe) {

if (this.singleEvent('Eat', item, null, foe, null, null)) {

this.runEvent('EatItem', foe, null, null, item);

if (item.id === 'leppaberry') foe.staleness = 'external';

}

if (item.onEat) foe.ateBerry = true;

};

} else if (item.fling.effect) {

move.onHit = item.fling.effect;

} else {

if (!move.secondaries) move.secondaries = [];

if (item.fling.status) {

move.secondaries.push({status: item.fling.status});

} else if (item.fling.volatileStatus) {

move.secondaries.push({volatileStatus: item.fling.volatileStatus});

}

}

source.setItem('');

source.lastItem = item.id;

source.usedItemThisTurn = true;

this.add("-enditem", source, item.name, '[from] move: Fling');

this.runEvent('AfterUseItem', source, null, null, item);

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Cute",

},

flipturn: {

num: 812,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "If this move is successful and the user has not fainted, the user switches out even if it is trapped and is replaced immediately by a selected party member. The user does not switch out if there are no unfainted party members, or if the target switched out using an Eject Button or through the effect of the Emergency Exit or Wimp Out Abilities.",

shortDesc: "User switches out after damaging the target.",

name: "Flip Turn",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

selfSwitch: true,

secondary: null,

target: "normal",

type: "Water",

},

floatyfall: {

num: 731,

accuracy: 95,

basePower: 90,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

isNonstandard: "LGPE",

name: "Floaty Fall",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, gravity: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Flying",

contestType: "Cool",

},

floralhealing: {

num: 666,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target restores 1/2 of its maximum HP, rounded half up. If the terrain is Grassy Terrain, the target instead restores 2/3 of its maximum HP, rounded half down.",

shortDesc: "Heals the target by 50% of its max HP.",

name: "Floral Healing",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, heal: 1, mystery: 1},

onHit(target, source) {

let success = false;

if (this.field.isTerrain('grassyterrain')) {

success = !!this.heal(this.modify(target.baseMaxhp, 0.667)); // TODO: find out the real value

} else {

success = !!this.heal(Math.ceil(target.baseMaxhp \* 0.5));

}

if (success && target.side !== source.side) {

target.staleness = 'external';

}

return success;

},

secondary: null,

target: "normal",

type: "Fairy",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

flowershield: {

num: 579,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the Defense of all active Grass-type Pokemon by 1 stage. Fails if there are no active Grass-type Pokemon.",

shortDesc: "Raises Defense by 1 of all active Grass types.",

name: "Flower Shield",

pp: 10,

priority: 0,

flags: {distance: 1},

onHitField(t, source, move) {

const targets: Pokemon[] = [];

for (const pokemon of this.getAllActive()) {

if (pokemon.hasType('Grass')) {

// This move affects every Grass-type Pokemon in play.

targets.push(pokemon);

}

}

let success = false;

for (const target of targets) {

success = this.boost({def: 1}, target, source, move) || success;

}

return success;

},

secondary: null,

target: "all",

type: "Fairy",

zMove: {boost: {def: 1}},

contestType: "Beautiful",

},

fly: {

num: 19,

accuracy: 95,

basePower: 90,

category: "Physical",

desc: "This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks other than Gust, Hurricane, Sky Uppercut, Smack Down, Thousand Arrows, Thunder, and Twister, and Gust and Twister have doubled power when used against it. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Flies up on first turn, then strikes the next turn.",

name: "Fly",

pp: 15,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1, gravity: 1, distance: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onInvulnerability(target, source, move) {

if (['gust', 'twister', 'skyuppercut', 'thunder', 'hurricane', 'smackdown', 'thousandarrows'].includes(move.id)) {

return;

}

return false;

},

onSourceModifyDamage(damage, source, target, move) {

if (move.id === 'gust' || move.id === 'twister') {

return this.chainModify(2);

}

},

},

secondary: null,

target: "any",

type: "Flying",

contestType: "Clever",

},

flyingpress: {

num: 560,

accuracy: 95,

basePower: 100,

category: "Physical",

desc: "This move combines Flying in its type effectiveness against the target. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "Combines Flying in its type effectiveness.",

name: "Flying Press",

pp: 10,

flags: {contact: 1, protect: 1, mirror: 1, gravity: 1, distance: 1, nonsky: 1},

onEffectiveness(typeMod, target, type, move) {

return typeMod + this.dex.getEffectiveness('Flying', type);

},

priority: 0,

secondary: null,

target: "any",

type: "Fighting",

zMove: {basePower: 170},

contestType: "Tough",

},

focusblast: {

num: 411,

accuracy: 70,

basePower: 120,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Focus Blast",

pp: 5,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Fighting",

contestType: "Cool",

},

focusenergy: {

num: 116,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's chance for a critical hit by 2 stages. Fails if the user already has the effect. Baton Pass can be used to transfer this effect to an ally.",

shortDesc: "Raises the user's critical hit ratio by 2.",

name: "Focus Energy",

pp: 30,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'focusenergy',

condition: {

onStart(target, source, effect) {

if (effect?.id === 'zpower') {

this.add('-start', target, 'move: Focus Energy', '[zeffect]');

} else if (effect && (['imposter', 'psychup', 'transform'].includes(effect.id))) {

this.add('-start', target, 'move: Focus Energy', '[silent]');

} else {

this.add('-start', target, 'move: Focus Energy');

}

},

onModifyCritRatio(critRatio) {

return critRatio + 2;

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {accuracy: 1}},

contestType: "Cool",

},

focuspunch: {

num: 264,

accuracy: 100,

basePower: 150,

category: "Physical",

desc: "The user loses its focus and does nothing if it is hit by a damaging attack this turn before it can execute the move.",

shortDesc: "Fails if the user takes damage before it hits.",

name: "Focus Punch",

pp: 20,

priority: -3,

flags: {contact: 1, protect: 1, punch: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('focuspunch');

},

beforeMoveCallback(pokemon) {

if (pokemon.volatiles['focuspunch'] && pokemon.volatiles['focuspunch'].lostFocus) {

this.add('cant', pokemon, 'Focus Punch', 'Focus Punch');

return true;

}

},

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'move: Focus Punch');

},

onHit(pokemon, source, move) {

if (move.category !== 'Status') {

pokemon.volatiles['focuspunch'].lostFocus = true;

}

},

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

followme: {

num: 266,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the turn, all single-target attacks from the opposing side are redirected to the user. Such attacks are redirected to the user before they can be reflected by Magic Coat or the Magic Bounce Ability, or drawn in by the Lightning Rod or Storm Drain Abilities. Fails if it is not a Double Battle or Battle Royal. This effect is ignored while the user is under the effect of Sky Drop.",

shortDesc: "The foes' moves target the user on the turn used.",

name: "Follow Me",

pp: 20,

priority: 2,

flags: {},

volatileStatus: 'followme',

onTryHit(target) {

if (target.side.active.length < 2) return false;

},

condition: {

duration: 1,

onStart(target, source, effect) {

if (effect?.id === 'zpower') {

this.add('-singleturn', target, 'move: Follow Me', '[zeffect]');

} else {

this.add('-singleturn', target, 'move: Follow Me');

}

},

onFoeRedirectTargetPriority: 1,

onFoeRedirectTarget(target, source, source2, move) {

if (!this.effectData.target.isSkyDropped() && this.validTarget(this.effectData.target, source, move.target)) {

if (move.smartTarget) move.smartTarget = false;

this.debug("Follow Me redirected target of move");

return this.effectData.target;

}

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

forcepalm: {

num: 395,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze the target.",

name: "Force Palm",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Fighting",

contestType: "Cool",

},

foresight: {

num: 193,

accuracy: true,

basePower: 0,

category: "Status",

desc: "As long as the target remains active, its evasiveness stat stage is ignored during accuracy checks against it if it is greater than 0, and Normal- and Fighting-type attacks can hit the target if it is a Ghost type. Fails if the target is already affected, or affected by Miracle Eye or Odor Sleuth.",

shortDesc: "Fighting, Normal hit Ghost. Evasiveness ignored.",

isNonstandard: "Past",

name: "Foresight",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'foresight',

onTryHit(target) {

if (target.volatiles['miracleeye']) return false;

},

condition: {

noCopy: true,

onStart(pokemon) {

this.add('-start', pokemon, 'Foresight');

},

onNegateImmunity(pokemon, type) {

if (pokemon.hasType('Ghost') && ['Normal', 'Fighting'].includes(type)) return false;

},

onModifyBoost(boosts) {

if (boosts.evasion && boosts.evasion > 0) {

boosts.evasion = 0;

}

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'crit2'},

contestType: "Clever",

},

forestscurse: {

num: 571,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the Grass type to be added to the target, effectively making it have two or three types. Fails if the target is already a Grass type. If Trick-or-Treat adds a type to the target, it replaces the type added by this move and vice versa.",

shortDesc: "Adds Grass to the target's type(s).",

name: "Forest's Curse",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target) {

if (target.hasType('Grass')) return false;

if (!target.addType('Grass')) return false;

this.add('-start', target, 'typeadd', 'Grass', '[from] move: Forest\'s Curse');

},

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Clever",

},

foulplay: {

num: 492,

accuracy: 100,

basePower: 95,

category: "Physical",

desc: "Damage is calculated using the target's Attack stat, including stat stage changes. The user's Ability, item, and burn are used as normal.",

shortDesc: "Uses target's Attack stat in damage calculation.",

name: "Foul Play",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

useTargetOffensive: true,

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

freezedry: {

num: 573,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "Has a 10% chance to freeze the target. This move's type effectiveness against Water is changed to be super effective no matter what this move's type is.",

shortDesc: "10% chance to freeze. Super effective on Water.",

name: "Freeze-Dry",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

onEffectiveness(typeMod, target, type) {

if (type === 'Water') return 1;

},

secondary: {

chance: 10,

status: 'frz',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

freezeshock: {

num: 553,

accuracy: 90,

basePower: 140,

category: "Physical",

desc: "Has a 30% chance to paralyze the target. This attack charges on the first turn and executes on the second. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Charges turn 1. Hits turn 2. 30% paralyze.",

name: "Freeze Shock",

pp: 5,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

freezyfrost: {

num: 739,

accuracy: 90,

basePower: 100,

category: "Special",

desc: "Resets the stat stages of all active Pokemon to 0.",

shortDesc: "Eliminates all stat changes.",

isNonstandard: "LGPE",

name: "Freezy Frost",

pp: 10,

priority: 0,

flags: {protect: 1},

onHit() {

this.add('-clearallboost');

for (const pokemon of this.getAllActive()) {

pokemon.clearBoosts();

}

},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Clever",

},

frenzyplant: {

num: 338,

accuracy: 90,

basePower: 150,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Frenzy Plant",

pp: 5,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1, nonsky: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

frostbreath: {

num: 524,

accuracy: 90,

basePower: 60,

category: "Special",

desc: "This move is always a critical hit unless the target is under the effect of Lucky Chant or has the Battle Armor or Shell Armor Abilities.",

shortDesc: "Always results in a critical hit.",

name: "Frost Breath",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

willCrit: true,

secondary: null,

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

frustration: {

num: 218,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon) {

return Math.floor(((255 - pokemon.happiness) \* 10) / 25) || 1;

},

category: "Physical",

desc: "Power is equal to the greater of ((255 - user's Happiness) \* 2/5), rounded down, or 1.",

shortDesc: "Max 102 power at minimum Happiness.",

isNonstandard: "Past",

name: "Frustration",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cute",

},

furyattack: {

num: 31,

accuracy: 85,

basePower: 15,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Fury Attack",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

furycutter: {

num: 210,

accuracy: 95,

basePower: 40,

basePowerCallback(pokemon, target, move) {

if (!pokemon.volatiles.furycutter || move.hit === 1) {

pokemon.addVolatile('furycutter');

}

return this.clampIntRange(move.basePower \* pokemon.volatiles.furycutter.multiplier, 1, 160);

},

category: "Physical",

desc: "Power doubles with each successful hit, up to a maximum of 160 power. The power is reset if this move misses or another move is used.",

shortDesc: "Power doubles with each hit, up to 160.",

name: "Fury Cutter",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

condition: {

duration: 2,

onStart() {

this.effectData.multiplier = 1;

},

onRestart() {

if (this.effectData.multiplier < 4) {

this.effectData.multiplier <<= 1;

}

this.effectData.duration = 2;

},

},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cool",

},

furyswipes: {

num: 154,

accuracy: 80,

basePower: 18,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Fury Swipes",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

maxMove: {basePower: 100},

contestType: "Tough",

},

fusionbolt: {

num: 559,

accuracy: 100,

basePower: 100,

category: "Physical",

desc: "Power doubles if the last move used by any Pokemon this turn was Fusion Flare.",

shortDesc: "Power doubles if used after Fusion Flare this turn.",

name: "Fusion Bolt",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower, pokemon) {

if (this.lastSuccessfulMoveThisTurn === 'fusionflare') {

this.debug('double power');

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

fusionflare: {

num: 558,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "Power doubles if the last move used by any Pokemon this turn was Fusion Bolt.",

shortDesc: "Power doubles if used after Fusion Bolt this turn.",

name: "Fusion Flare",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

onBasePower(basePower, pokemon) {

if (this.lastSuccessfulMoveThisTurn === 'fusionbolt') {

this.debug('double power');

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

futuresight: {

num: 248,

accuracy: 100,

basePower: 120,

category: "Special",

desc: "Deals damage two turns after this move is used. At the end of that turn, the damage is calculated at that time and dealt to the Pokemon at the position the target had when the move was used. If the user is no longer active at the time, damage is calculated based on the user's natural Special Attack stat, types, and level, with no boosts from its held item or Ability. Fails if this move or Doom Desire is already in effect for the target's position.",

shortDesc: "Hits two turns after being used.",

name: "Future Sight",

pp: 10,

priority: 0,

flags: {},

ignoreImmunity: true,

isFutureMove: true,

onTry(source, target) {

if (!target.side.addSlotCondition(target, 'futuremove')) return false;

Object.assign(target.side.slotConditions[target.position]['futuremove'], {

duration: 3,

move: 'futuresight',

source: source,

moveData: {

id: 'futuresight',

name: "Future Sight",

accuracy: 100,

basePower: 120,

category: "Special",

priority: 0,

flags: {},

ignoreImmunity: false,

effectType: 'Move',

isFutureMove: true,

type: 'Psychic',

},

});

this.add('-start', source, 'move: Future Sight');

return null;

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

gastroacid: {

num: 380,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target's Ability to be rendered ineffective as long as it remains active. If the target uses Baton Pass, the replacement will remain under this effect. If the target's Ability is Battle Bond, Comatose, Disguise, Multitype, Power Construct, RKS System, Schooling, Shields Down, Stance Change, or Zen Mode, this move fails, and receiving the effect through Baton Pass ends the effect immediately.",

shortDesc: "Nullifies the target's Ability.",

name: "Gastro Acid",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

volatileStatus: 'gastroacid',

onTryHit(pokemon) {

const bannedAbilities = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'zenmode',

];

if (bannedAbilities.includes(pokemon.ability)) {

return false;

}

},

condition: {

// Ability suppression implemented in Pokemon.ignoringAbility() within sim/pokemon.js

onStart(pokemon) {

this.add('-endability', pokemon);

this.singleEvent('End', pokemon.getAbility(), pokemon.abilityData, pokemon, pokemon, 'gastroacid');

},

},

secondary: null,

target: "normal",

type: "Poison",

zMove: {boost: {spe: 1}},

contestType: "Tough",

},

geargrind: {

num: 544,

accuracy: 85,

basePower: 50,

category: "Physical",

desc: "Hits twice. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times in one turn.",

name: "Gear Grind",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 2,

secondary: null,

target: "normal",

type: "Steel",

zMove: {basePower: 180},

maxMove: {basePower: 130},

contestType: "Clever",

},

gearup: {

num: 674,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the Attack and Special Attack of Pokemon on the user's side with the Plus or Minus Abilities by 1 stage.",

shortDesc: "Raises Atk, Sp. Atk of allies with Plus/Minus by 1.",

name: "Gear Up",

pp: 20,

priority: 0,

flags: {snatch: 1, authentic: 1},

onHitSide(side, source, move) {

const targets = [];

for (const pokemon of side.active) {

if (pokemon.hasAbility(['plus', 'minus'])) {

targets.push(pokemon);

}

}

if (!targets.length) return false;

let didSomething = false;

for (const target of targets) {

didSomething = this.boost({atk: 1, spa: 1}, target, source, move, false, true) || didSomething;

}

return didSomething;

},

secondary: null,

target: "allySide",

type: "Steel",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

genesissupernova: {

num: 703,

accuracy: true,

basePower: 185,

category: "Special",

desc: "If this move is successful, the terrain becomes Psychic Terrain.",

shortDesc: "Summons Psychic Terrain.",

isNonstandard: "Past",

name: "Genesis Supernova",

pp: 1,

priority: 0,

flags: {},

isZ: "mewniumz",

secondary: {

chance: 100,

self: {

onHit() {

this.field.setTerrain('psychicterrain');

},

},

},

target: "normal",

type: "Psychic",

contestType: "Cool",

},

geomancy: {

num: 601,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Attack, Special Defense, and Speed by 2 stages. This attack charges on the first turn and executes on the second. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Charges, then raises SpA, SpD, Spe by 2 turn 2.",

isNonstandard: "Past",

name: "Geomancy",

pp: 10,

priority: 0,

flags: {charge: 1, nonsky: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

boosts: {

spa: 2,

spd: 2,

spe: 2,

},

secondary: null,

target: "self",

type: "Fairy",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Beautiful",

},

gigadrain: {

num: 202,

accuracy: 100,

basePower: 75,

category: "Special",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Giga Drain",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Grass",

contestType: "Clever",

},

gigaimpact: {

num: 416,

accuracy: 90,

basePower: 150,

category: "Physical",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Giga Impact",

pp: 5,

priority: 0,

flags: {contact: 1, recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

gigavolthavoc: {

num: 646,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Gigavolt Havoc",

pp: 1,

priority: 0,

flags: {},

isZ: "electriumz",

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

glaciate: {

num: 549,

accuracy: 95,

basePower: 65,

category: "Special",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Speed by 1.",

name: "Glaciate",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "allAdjacentFoes",

type: "Ice",

contestType: "Beautiful",

},

glare: {

num: 137,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Paralyzes the target.",

shortDesc: "Paralyzes the target.",

name: "Glare",

pp: 30,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'par',

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Tough",

},

glitzyglow: {

num: 736,

accuracy: 95,

basePower: 80,

category: "Special",

desc: "This move summons Light Screen for 5 turns upon use.",

shortDesc: "Summons Light Screen.",

isNonstandard: "LGPE",

name: "Glitzy Glow",

pp: 15,

priority: 0,

flags: {protect: 1},

self: {

sideCondition: 'lightscreen',

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

gmaxbefuddle: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side either falls asleep, becomes poisoned, or becomes paralyzed, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: slp or psn or par.",

isNonstandard: "Gigantamax",

name: "G-Max Befuddle",

pp: 5,

priority: 0,

flags: {},

isMax: "Butterfree",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

const result = this.random(3);

if (result === 0) {

pokemon.trySetStatus('slp', source);

} else if (result === 1) {

pokemon.trySetStatus('par', source);

} else {

pokemon.trySetStatus('psn', source);

}

}

},

},

target: "adjacentFoe",

type: "Bug",

contestType: "Cool",

},

gmaxcannonade: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, for 4 turns each non-Water-type Pokemon on the opposing side takes damage equal to 1/6 of its maximum HP, rounded down, at the end of each turn during effect, including the last turn.",

shortDesc: "Base move affects power. Foes: -1/6 HP, 4 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Cannonade",

pp: 10,

priority: 0,

flags: {},

isMax: "Blastoise",

self: {

onHit(source) {

source.side.foe.addSideCondition('gmaxcannonade');

},

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'G-Max Cannonade');

},

onResidualOrder: 5,

onResidualSubOrder: 1.1,

onResidual(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Water')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

},

onEnd(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Water')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

this.add('-sideend', targetSide, 'G-Max Cannonade');

},

},

secondary: null,

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

gmaxcentiferno: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side is prevented from switching for four or five turns (seven turns if the user is holding Grip Claw), even if they have a substitute. Causes damage equal to 1/8 of their maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. They can still switch out if they are holding Shed Shell or use Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends for a target if it leaves the field, or if it uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Base move affects power. Foes: bound 4-5 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Centiferno",

pp: 5,

priority: 0,

flags: {},

isMax: "Centiskorch",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('partiallytrapped', source, this.dex.getActiveMove('G-Max Centiferno'));

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Fire",

contestType: "Cool",

},

gmaxchistrike: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the user's side has their critical hit ratio raised by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: Crit Ratio +1.",

isNonstandard: "Gigantamax",

name: "G-Max Chi Strike",

pp: 5,

priority: 0,

flags: {},

isMax: "Machamp",

self: {

onHit(source) {

for (const pokemon of source.side.active) {

pokemon.addVolatile('gmaxchistrike');

}

},

},

condition: {

noCopy: true,

onStart(target, source, effect) {

this.effectData.layers = 1;

if (!['imposter', 'psychup', 'transform'].includes(effect?.id)) {

this.add('-start', target, 'move: G-Max Chi Strike');

}

},

onRestart(target, source, effect) {

if (this.effectData.layers >= 3) return false;

this.effectData.layers++;

if (!['imposter', 'psychup', 'transform'].includes(effect?.id)) {

this.add('-start', target, 'move: G-Max Chi Strike');

}

},

onModifyCritRatio(critRatio) {

return critRatio + this.effectData.layers;

},

},

secondary: null,

target: "adjacentFoe",

type: "Fighting",

contestType: "Cool",

},

gmaxcuddle: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side becomes infatuated, even if they have a substitute. This effect does not happen for a target if both it and the user are the same gender, if either is genderless, or if the target is already infatuated.",

shortDesc: "Base move affects power. Foes: infatuated.",

isNonstandard: "Gigantamax",

name: "G-Max Cuddle",

pp: 5,

priority: 0,

flags: {},

isMax: "Eevee",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('attract');

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Normal",

contestType: "Cool",

},

gmaxdepletion: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side loses 2 PP from its last move used, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: last move -2 PP.",

isNonstandard: "Gigantamax",

name: "G-Max Depletion",

pp: 5,

priority: 0,

flags: {},

isMax: "Duraludon",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

const move = pokemon.lastMove;

if (move && !move.isZ && !move.isMax) {

const ppDeducted = pokemon.deductPP(move.id, 4);

if (ppDeducted) {

this.add("-activate", pokemon, 'move: Max Depletion', move.name, ppDeducted);

return;

}

}

return false;

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Dragon",

contestType: "Cool",

},

gmaxdrumsolo: {

num: 1000,

accuracy: true,

basePower: 160,

category: "Physical",

desc: "This move will always have 160 Base Power, and it ignores the abilities of opposing Pokemon.",

shortDesc: "Stronger than other Max Moves. Ignores abilities.",

isNonstandard: "Gigantamax",

name: "G-Max Drum Solo",

pp: 5,

priority: 0,

flags: {},

isMax: "Rillaboom",

ignoreAbility: true,

secondary: null,

target: "adjacentFoe",

type: "Grass",

contestType: "Cool",

},

gmaxfinale: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the user's side restores 1/6 of its current maximum HP, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: +1/6 max HP.",

isNonstandard: "Gigantamax",

name: "G-Max Finale",

pp: 5,

priority: 0,

flags: {},

isMax: "Alcremie",

self: {

onHit(target, source, move) {

for (const pokemon of source.side.active) {

this.heal(pokemon.maxhp / 6, pokemon, source, move);

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Fairy",

contestType: "Cool",

},

gmaxfireball: {

num: 1000,

accuracy: true,

basePower: 160,

category: "Physical",

desc: "This move will always have 160 Base Power, and it ignores the abilities of opposing Pokemon.",

shortDesc: "Stronger than other Max Moves. Ignores abilities.",

isNonstandard: "Gigantamax",

name: "G-Max Fireball",

pp: 5,

priority: 0,

flags: {},

isMax: "Cinderace",

ignoreAbility: true,

secondary: null,

target: "adjacentFoe",

type: "Fire",

contestType: "Cool",

},

gmaxfoamburst: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Speed of each Pokemon on the opposing side is lowered by 2 stages, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -2 Speed.",

isNonstandard: "Gigantamax",

name: "G-Max Foam Burst",

pp: 5,

priority: 0,

flags: {},

isMax: "Kingler",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

this.boost({spe: -2}, pokemon);

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

gmaxgoldrush: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side becomes confused, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: confused.",

isNonstandard: "Gigantamax",

name: "G-Max Gold Rush",

pp: 5,

priority: 0,

flags: {},

isMax: "Meowth",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('confusion');

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Normal",

contestType: "Cool",

},

gmaxgravitas: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Gravity begins.",

shortDesc: "Base move affects power. Starts Gravity.",

isNonstandard: "Gigantamax",

name: "G-Max Gravitas",

pp: 5,

priority: 0,

flags: {},

isMax: "Orbeetle",

self: {

pseudoWeather: 'gravity',

},

target: "adjacentFoe",

type: "Psychic",

contestType: "Cool",

},

gmaxhydrosnipe: {

num: 1000,

accuracy: true,

basePower: 160,

category: "Physical",

desc: "This move will always have 160 Base Power, and it ignores the abilities of opposing Pokemon.",

shortDesc: "Stronger than other Max Moves. Ignores abilities.",

isNonstandard: "Gigantamax",

name: "G-Max Hydrosnipe",

pp: 5,

priority: 0,

flags: {},

isMax: "Inteleon",

ignoreAbility: true,

secondary: null,

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

gmaxmalodor: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side becomes poisoned, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: poisoned.",

isNonstandard: "Gigantamax",

name: "G-Max Malodor",

pp: 5,

priority: 0,

flags: {},

isMax: "Garbodor",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.trySetStatus('psn', source);

}

},

},

target: "adjacentFoe",

type: "Poison",

contestType: "Cool",

},

gmaxmeltdown: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Torment begins for each Pokemon on the opposing side, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: Tormented.",

isNonstandard: "Gigantamax",

name: "G-Max Meltdown",

pp: 5,

priority: 0,

flags: {},

isMax: "Melmetal",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

if (!pokemon.volatiles['dynamax']) pokemon.addVolatile('torment');

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Steel",

contestType: "Cool",

},

gmaxoneblow: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful and any Pokemon on the opposing side is using Baneful Bunker, Detect, King's Shield, Mat Block, Max Guard, Obstruct, Protect, or Spiky Shield, this move will fully break the protection.",

shortDesc: "Base move affects power. Breaks all protection.",

isNonstandard: "Gigantamax",

name: "G-Max One Blow",

pp: 5,

priority: 0,

flags: {},

isMax: "Urshifu",

breaksProtect: true,

secondary: null,

target: "adjacentFoe",

type: "Dark",

contestType: "Cool",

},

gmaxrapidflow: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful and any Pokemon on the opposing side is using Baneful Bunker, Detect, King's Shield, Mat Block, Max Guard, Obstruct, Protect, or Spiky Shield, this move will fully break the protection.",

shortDesc: "Base move affects power. Breaks all protection.",

isNonstandard: "Gigantamax",

name: "G-Max Rapid Flow",

pp: 5,

priority: 0,

flags: {},

isMax: "Urshifu-Rapid-Strike",

breaksProtect: true,

secondary: null,

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

gmaxreplenish: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, there is a 50% chance every Pokemon on the user's side has its Berry restored, even if they have a substitute.",

shortDesc: "Base move affects power. 50% restores Berries.",

isNonstandard: "Gigantamax",

name: "G-Max Replenish",

pp: 5,

priority: 0,

flags: {},

isMax: "Snorlax",

self: {

onHit(source) {

if (this.random(2) === 0) return;

for (const pokemon of source.side.active) {

if (!pokemon.item && pokemon.lastItem && this.dex.getItem(pokemon.lastItem).isBerry) {

const item = pokemon.lastItem;

pokemon.lastItem = '';

this.add('-item', pokemon, this.dex.getItem(item), '[from] move: G-Max Replenish');

pokemon.setItem(item);

}

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Normal",

contestType: "Cool",

},

gmaxresonance: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Aurora Veil begins on the user's side.",

shortDesc: "Base move affects power. Allies: Aurora Veil.",

isNonstandard: "Gigantamax",

name: "G-Max Resonance",

pp: 5,

priority: 0,

flags: {},

isMax: "Lapras",

self: {

sideCondition: 'auroraveil',

},

secondary: null,

target: "adjacentFoe",

type: "Ice",

contestType: "Cool",

},

gmaxsandblast: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side is prevented from switching for four or five turns (seven turns if the user is holding Grip Claw), even if they have a substitute. Causes damage equal to 1/8 of their maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. They can still switch out if they are holding Shed Shell or use Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends for a target if it leaves the field, or if it uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Base move affects power. Foes: bound 4-5 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Sandblast",

pp: 5,

priority: 0,

flags: {},

isMax: "Sandaconda",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('partiallytrapped', source, this.dex.getActiveMove('G-Max Sandblast'));

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Ground",

contestType: "Cool",

},

gmaxsmite: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side becomes confused, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: confused.",

isNonstandard: "Gigantamax",

name: "G-Max Smite",

pp: 5,

priority: 0,

flags: {},

isMax: "Hatterene",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('confusion', source);

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Fairy",

contestType: "Cool",

},

gmaxsnooze: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, there is a 50% chance the effect of Yawn begins on the target, even if it has a substitute.",

shortDesc: "Base move affects power. Target: 50% Yawn.",

isNonstandard: "Gigantamax",

name: "G-Max Snooze",

pp: 5,

priority: 0,

flags: {},

isMax: "Grimmsnarl",

onHit(target) {

if (target.status || !target.runStatusImmunity('slp')) return;

if (this.random(2) === 0) return;

target.addVolatile('yawn');

},

onAfterSubDamage(damage, target) {

if (target.status || !target.runStatusImmunity('slp')) return;

if (this.random(2) === 0) return;

target.addVolatile('yawn');

},

secondary: null,

target: "adjacentFoe",

type: "Dark",

contestType: "Cool",

},

gmaxsteelsurge: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, it sets up a hazard on the opposing side of the field, damaging each opposing Pokemon that switches in. Foes lose 1/32, 1/16, 1/8, 1/4, or 1/2 of their maximum HP, rounded down, based on their weakness to the Steel type; 0.25x, 0.5x, neutral, 2x, or 4x, respectively. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, or is hit by Defog.",

shortDesc: "Base move affects power. Foes: Steel hazard.",

isNonstandard: "Gigantamax",

name: "G-Max Steelsurge",

pp: 5,

priority: 0,

flags: {},

isMax: "Copperajah",

self: {

onHit(source) {

source.side.foe.addSideCondition('gmaxsteelsurge');

},

},

condition: {

onStart(side) {

this.add('-sidestart', side, 'move: G-Max Steelsurge');

},

onSwitchIn(pokemon) {

if (pokemon.hasItem('heavydutyboots')) return;

// Ice Face and Disguise correctly get typed damage from Stealth Rock

// because Stealth Rock bypasses Substitute.

// They don't get typed damage from Steelsurge because Steelsurge doesn't,

// so we're going to test the damage of a Steel-type Stealth Rock instead.

const steelHazard = this.dex.getActiveMove('Stealth Rock');

steelHazard.type = 'Steel';

const typeMod = this.clampIntRange(pokemon.runEffectiveness(steelHazard), -6, 6);

this.damage(pokemon.maxhp \* Math.pow(2, typeMod) / 8);

},

},

secondary: null,

target: "adjacentFoe",

type: "Steel",

contestType: "Cool",

},

gmaxstonesurge: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, it sets up a hazard on the opposing side of the field, damaging each opposing Pokemon that switches in. Foes lose 1/32, 1/16, 1/8, 1/4, or 1/2 of their maximum HP, rounded down, based on their weakness to the Rock type; 0.25x, 0.5x, neutral, 2x, or 4x, respectively. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, or is hit by Defog.",

shortDesc: "Base move affects power. Foes: Stealth Rock.",

isNonstandard: "Gigantamax",

name: "G-Max Stonesurge",

pp: 5,

priority: 0,

flags: {},

isMax: "Drednaw",

self: {

onHit(source) {

source.side.foe.addSideCondition('stealthrock');

},

},

secondary: null,

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

gmaxstunshock: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side either becomes poisoned or paralyzed, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: psn or par.",

isNonstandard: "Gigantamax",

name: "G-Max Stun Shock",

pp: 10,

priority: 0,

flags: {},

isMax: "Toxtricity",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

const result = this.random(2);

if (result === 0) {

pokemon.trySetStatus('par', source);

} else {

pokemon.trySetStatus('psn', source);

}

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Electric",

contestType: "Cool",

},

gmaxsweetness: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the user's side has its status condition cured, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: status cured.",

isNonstandard: "Gigantamax",

name: "G-Max Sweetness",

pp: 10,

priority: 0,

flags: {},

isMax: "Appletun",

self: {

onHit(source) {

for (const ally of source.side.pokemon) {

ally.cureStatus();

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Grass",

contestType: "Cool",

},

gmaxtartness: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the evasiveness of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 evasiveness.",

isNonstandard: "Gigantamax",

name: "G-Max Tartness",

pp: 10,

priority: 0,

flags: {},

isMax: "Flapple",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

this.boost({evasion: -1}, pokemon);

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Grass",

contestType: "Cool",

},

gmaxterror: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side is prevented from switching out, even if they have a substitute. They can still switch out if they are holding Shed Shell or use Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If a target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Base move affects power. Foes: trapped.",

isNonstandard: "Gigantamax",

name: "G-Max Terror",

pp: 10,

priority: 0,

flags: {},

isMax: "Gengar",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.addVolatile('trapped', source, null, 'trapper');

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Ghost",

contestType: "Cool",

},

gmaxvinelash: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, for 4 turns each non-Grass-type Pokemon on the opposing side takes damage equal to 1/6 of its maximum HP, rounded down, at the end of each turn during effect, including the last turn.",

shortDesc: "Base move affects power. Foes: -1/6 HP, 4 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Vine Lash",

pp: 10,

priority: 0,

flags: {},

isMax: "Venusaur",

self: {

onHit(source) {

source.side.foe.addSideCondition('gmaxvinelash');

},

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'G-Max Vine Lash');

},

onResidualOrder: 5,

onResidualSubOrder: 1.1,

onResidual(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Grass')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

},

onEnd(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Grass')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

this.add('-sideend', targetSide, 'G-Max Vine Lash');

},

},

secondary: null,

target: "adjacentFoe",

type: "Grass",

contestType: "Cool",

},

gmaxvolcalith: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, for 4 turns each non-Rock-type Pokemon on the opposing side takes damage equal to 1/6 of its maximum HP, rounded down, at the end of each turn during effect, including the last turn.",

shortDesc: "Base move affects power. Foes: -1/6 HP, 4 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Volcalith",

pp: 10,

priority: 0,

flags: {},

isMax: "Coalossal",

self: {

onHit(source) {

source.side.foe.addSideCondition('gmaxvolcalith');

},

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'G-Max Volcalith');

},

onResidualOrder: 5,

onResidualSubOrder: 1.1,

onResidual(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Rock')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

},

onEnd(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Rock')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

this.add('-sideend', targetSide, 'G-Max Volcalith');

},

},

secondary: null,

target: "adjacentFoe",

type: "Rock",

contestType: "Cool",

},

gmaxvoltcrash: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, each Pokemon on the opposing side becomes paralyzed, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: paralyzed.",

isNonstandard: "Gigantamax",

name: "G-Max Volt Crash",

pp: 10,

priority: 0,

flags: {},

isMax: "Pikachu",

self: {

onHit(source) {

for (const pokemon of source.side.foe.active) {

pokemon.trySetStatus('par', source);

}

},

},

secondary: null,

target: "adjacentFoe",

type: "Electric",

contestType: "Cool",

},

gmaxwildfire: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, for 4 turns each non-Fire-type Pokemon on the opposing side takes damage equal to 1/6 of its maximum HP, rounded down, at the end of each turn during effect, including the last turn.",

shortDesc: "Base move affects power. Foes: -1/6 HP, 4 turns.",

isNonstandard: "Gigantamax",

name: "G-Max Wildfire",

pp: 10,

priority: 0,

flags: {},

isMax: "Charizard",

self: {

onHit(source) {

source.side.foe.addSideCondition('gmaxwildfire');

},

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'G-Max Wildfire');

},

onResidualOrder: 5,

onResidualSubOrder: 1.1,

onResidual(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Fire')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

},

onEnd(targetSide) {

for (const pokemon of targetSide.active) {

if (!pokemon.hasType('Fire')) this.damage(pokemon.baseMaxhp / 6, pokemon);

}

this.add('-sideend', targetSide, 'G-Max Wildfire');

},

},

secondary: null,

target: "adjacentFoe",

type: "Fire",

contestType: "Cool",

},

gmaxwindrage: {

num: 1000,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effects of Electric Terrain, Grassy Terrain, Misty Terrain, and Psychic Terrain end, the effects of Reflect, Light Screen, Aurora Veil, Safeguard, Mist, G-Max Steelsurge, Spikes, Toxic Spikes, Stealth Rock, and Sticky Web end for the target's side, and the effects of G-Max Steelsurge, Spikes, Toxic Spikes, Stealth Rock, and Sticky Web end for the user's side.",

shortDesc: "Base move affects power. Ends Terrain, hazards.",

isNonstandard: "Gigantamax",

name: "G-Max Wind Rage",

pp: 10,

priority: 0,

flags: {},

isMax: "Corviknight",

self: {

onHit(source) {

let success = false;

const removeTarget = [

'reflect', 'lightscreen', 'auroraveil', 'safeguard', 'mist', 'spikes', 'toxicspikes', 'stealthrock', 'stickyweb',

];

const removeAll = ['spikes', 'toxicspikes', 'stealthrock', 'stickyweb', 'gmaxsteelsurge'];

for (const targetCondition of removeTarget) {

if (source.side.foe.removeSideCondition(targetCondition)) {

if (!removeAll.includes(targetCondition)) continue;

this.add('-sideend', source.side.foe, this.dex.getEffect(targetCondition).name, '[from] move: G-Max Wind Rage', '[of] ' + source);

success = true;

}

}

for (const sideCondition of removeAll) {

if (source.side.removeSideCondition(sideCondition)) {

this.add('-sideend', source.side, this.dex.getEffect(sideCondition).name, '[from] move: G-Max Wind Rage', '[of] ' + source);

success = true;

}

}

this.field.clearTerrain();

return success;

},

},

secondary: null,

target: "adjacentFoe",

type: "Flying",

contestType: "Cool",

},

grassknot: {

num: 447,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const targetWeight = target.getWeight();

if (targetWeight >= 2000) {

this.debug('120 bp');

return 120;

}

if (targetWeight >= 1000) {

this.debug('100 bp');

return 100;

}

if (targetWeight >= 500) {

this.debug('80 bp');

return 80;

}

if (targetWeight >= 250) {

this.debug('60 bp');

return 60;

}

if (targetWeight >= 100) {

this.debug('40 bp');

return 40;

}

this.debug('20 bp');

return 20;

},

category: "Special",

desc: "This move's power is 20 if the target weighs less than 10 kg, 40 if less than 25 kg, 60 if less than 50 kg, 80 if less than 100 kg, 100 if less than 200 kg, and 120 if greater than or equal to 200 kg.",

shortDesc: "More power the heavier the target.",

name: "Grass Knot",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

onTryHit(target, source, move) {

if (target.volatiles['dynamax']) {

this.add('-fail', source, 'move: Grass Knot', '[from] Dynamax');

this.attrLastMove('[still]');

return null;

}

},

secondary: null,

target: "normal",

type: "Grass",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cute",

},

grasspledge: {

num: 520,

accuracy: 100,

basePower: 80,

basePowerCallback(target, source, move) {

if (['waterpledge', 'firepledge'].includes(move.sourceEffect)) {

this.add('-combine');

return 150;

}

return 80;

},

category: "Special",

desc: "If one of the user's allies chose to use Fire Pledge or Water Pledge this turn and has not moved yet, it takes its turn immediately after the user and the user's move does nothing. If combined with Fire Pledge, the ally uses Fire Pledge with 150 power and a sea of fire appears on the target's side for 4 turns, which causes damage to non-Fire types equal to 1/8 of their maximum HP, rounded down, at the end of each turn during effect, including the last turn. If combined with Water Pledge, the ally uses Grass Pledge with 150 power and a swamp appears on the target's side for 4 turns, which quarters the Speed of each Pokemon on that side. When used as a combined move, this move gains STAB no matter what the user's type is. This move does not consume the user's Grass Gem.",

shortDesc: "Use with Fire or Water Pledge for added effect.",

name: "Grass Pledge",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onPrepareHit(target, source, move) {

for (const action of this.queue) {

if (

// @ts-ignore

!action.move || !action.pokemon || !action.pokemon.isActive ||

// @ts-ignore

action.pokemon.fainted || action.maxMove || action.zmove

) {

continue;

}

// @ts-ignore

if (action.pokemon.side === source.side && ['waterpledge', 'firepledge'].includes(action.move.id)) {

// @ts-ignore

this.queue.prioritizeAction(action, move);

this.add('-waiting', source, action.pokemon);

return null;

}

}

},

onModifyMove(move) {

if (move.sourceEffect === 'waterpledge') {

move.type = 'Grass';

move.forceSTAB = true;

move.sideCondition = 'grasspledge';

}

if (move.sourceEffect === 'firepledge') {

move.type = 'Fire';

move.forceSTAB = true;

move.sideCondition = 'firepledge';

}

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'Grass Pledge');

},

onEnd(targetSide) {

this.add('-sideend', targetSide, 'Grass Pledge');

},

onModifySpe(spe, pokemon) {

return this.chainModify(0.25);

},

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Beautiful",

},

grasswhistle: {

num: 320,

accuracy: 55,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

isNonstandard: "Past",

name: "Grass Whistle",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

grassyglide: {

num: 803,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "If the current terrain is Grassy Terrain and the user is grounded, this move has its priority increased by 1.",

shortDesc: "User on Grassy Terrain: +1 priority.",

name: "Grassy Glide",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mystery: 1},

onModifyPriority(priority, source, target, move) {

if (this.field.isTerrain('grassyterrain') && source.isGrounded()) {

return priority + 1;

}

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

grassyterrain: {

num: 580,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the terrain becomes Grassy Terrain. During the effect, the power of Grass-type attacks used by grounded Pokemon is multiplied by 1.3, the power of Bulldoze, Earthquake, and Magnitude used against grounded Pokemon is multiplied by 0.5, and grounded Pokemon have 1/16 of their maximum HP, rounded down, restored at the end of each turn, including the last turn. Camouflage transforms the user into a Grass type, Nature Power becomes Energy Ball, and Secret Power has a 30% chance to cause sleep. Fails if the current terrain is Grassy Terrain.",

shortDesc: "5 turns. Grounded: +Grass power, +1/16 max HP.",

name: "Grassy Terrain",

pp: 10,

priority: 0,

flags: {nonsky: 1},

terrain: 'grassyterrain',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasItem('terrainextender')) {

return 8;

}

return 5;

},

onBasePowerPriority: 6,

onBasePower(basePower, attacker, defender, move) {

const weakenedMoves = ['earthquake', 'bulldoze', 'magnitude'];

if (weakenedMoves.includes(move.id)) {

this.debug('move weakened by grassy terrain');

return this.chainModify(0.5);

}

if (move.type === 'Grass' && attacker.isGrounded()) {

this.debug('grassy terrain boost');

return this.chainModify([0x14CD, 0x1000]);

}

},

onStart(battle, source, effect) {

if (effect?.effectType === 'Ability') {

this.add('-fieldstart', 'move: Grassy Terrain', '[from] ability: ' + effect, '[of] ' + source);

} else {

this.add('-fieldstart', 'move: Grassy Terrain');

}

},

onResidualOrder: 5,

onResidualSubOrder: 3,

onResidual() {

this.eachEvent('Terrain');

},

onTerrain(pokemon) {

if (pokemon.isGrounded() && !pokemon.isSemiInvulnerable()) {

this.debug('Pokemon is grounded, healing through Grassy Terrain.');

this.heal(pokemon.baseMaxhp / 16, pokemon, pokemon);

}

},

onEnd() {

if (!this.effectData.duration) this.eachEvent('Terrain');

this.add('-fieldend', 'move: Grassy Terrain');

},

},

secondary: null,

target: "all",

type: "Grass",

zMove: {boost: {def: 1}},

contestType: "Beautiful",

},

gravapple: {

num: 788,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 100% chance to lower the target's Defense by 1 stage. Power is multiplied by 1.5 during Gravity's effect.",

shortDesc: "Target: 100% -1 Def. During Gravity: 1.5x power.",

name: "Grav Apple",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower) {

if (this.field.getPseudoWeather('gravity')) {

return this.chainModify(1.5);

}

},

secondary: {

chance: 100,

boosts: {

def: -1,

},

},

target: "normal",

type: "Grass",

},

gravity: {

num: 356,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the evasiveness of all active Pokemon is multiplied by 0.6. At the time of use, Bounce, Fly, Magnet Rise, Sky Drop, and Telekinesis end immediately for all active Pokemon. During the effect, Bounce, Fly, Flying Press, High Jump Kick, Jump Kick, Magnet Rise, Sky Drop, Splash, and Telekinesis are prevented from being used by all active Pokemon. Ground-type attacks, Spikes, Toxic Spikes, Sticky Web, and the Arena Trap Ability can affect Flying types or Pokemon with the Levitate Ability. Fails if this move is already in effect.",

shortDesc: "5 turns: no Ground immunities, 1.67x accuracy.",

name: "Gravity",

pp: 5,

priority: 0,

flags: {nonsky: 1},

pseudoWeather: 'gravity',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onStart() {

this.add('-fieldstart', 'move: Gravity');

for (const pokemon of this.getAllActive()) {

let applies = false;

if (pokemon.removeVolatile('bounce') || pokemon.removeVolatile('fly')) {

applies = true;

this.queue.cancelMove(pokemon);

pokemon.removeVolatile('twoturnmove');

}

if (pokemon.volatiles['skydrop']) {

applies = true;

this.queue.cancelMove(pokemon);

if (pokemon.volatiles['skydrop'].source) {

this.add('-end', pokemon.volatiles['twoturnmove'].source, 'Sky Drop', '[interrupt]');

}

pokemon.removeVolatile('skydrop');

pokemon.removeVolatile('twoturnmove');

}

if (pokemon.volatiles['magnetrise']) {

applies = true;

delete pokemon.volatiles['magnetrise'];

}

if (pokemon.volatiles['telekinesis']) {

applies = true;

delete pokemon.volatiles['telekinesis'];

}

if (applies) this.add('-activate', pokemon, 'move: Gravity');

}

},

onModifyAccuracy(accuracy) {

if (typeof accuracy !== 'number') return;

return accuracy \* 5 / 3;

},

onDisableMove(pokemon) {

for (const moveSlot of pokemon.moveSlots) {

if (this.dex.getMove(moveSlot.id).flags['gravity']) {

pokemon.disableMove(moveSlot.id);

}

}

},

// groundedness implemented in battle.engine.js:BattlePokemon#isGrounded

onBeforeMovePriority: 6,

onBeforeMove(pokemon, target, move) {

if (move.flags['gravity']) {

this.add('cant', pokemon, 'move: Gravity', move);

return false;

}

},

onResidualOrder: 22,

onEnd() {

this.add('-fieldend', 'move: Gravity');

},

},

secondary: null,

target: "all",

type: "Psychic",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

growl: {

num: 45,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 1 stage.",

shortDesc: "Lowers the foe(s) Attack by 1.",

name: "Growl",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

boosts: {

atk: -1,

},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

growth: {

num: 74,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack and Special Attack by 1 stage. If the weather is Sunny Day or Desolate Land, this move raises the user's Attack and Special Attack by 2 stages. If the user is holding Utility Umbrella, this move will only raise the user's Attack and Special Attack by 1 stage, even if the weather is Sunny Day or Desolate Land.",

shortDesc: "Raises user's Attack and Sp. Atk by 1; 2 in Sun.",

name: "Growth",

pp: 20,

priority: 0,

flags: {snatch: 1},

onModifyMove(move, pokemon) {

if (['sunnyday', 'desolateland'].includes(pokemon.effectiveWeather())) move.boosts = {atk: 2, spa: 2};

},

boosts: {

atk: 1,

spa: 1,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {spa: 1}},

contestType: "Beautiful",

},

grudge: {

num: 288,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the user's next turn, if an opposing Pokemon's attack knocks the user out, that move loses all its remaining PP.",

shortDesc: "If the user faints, the attack used loses all its PP.",

name: "Grudge",

pp: 5,

priority: 0,

flags: {authentic: 1},

volatileStatus: 'grudge',

condition: {

onStart(pokemon) {

this.add('-singlemove', pokemon, 'Grudge');

},

onFaint(target, source, effect) {

if (!source || source.fainted || !effect) return;

if (effect.effectType === 'Move' && !effect.isFutureMove && source.lastMove) {

for (const moveSlot of source.moveSlots) {

if (moveSlot.id === source.lastMove.id) {

moveSlot.pp = 0;

this.add('-activate', source, 'move: Grudge', this.dex.getMove(source.lastMove.id).name);

}

}

}

},

onBeforeMovePriority: 100,

onBeforeMove(pokemon) {

this.debug('removing Grudge before attack');

pokemon.removeVolatile('grudge');

},

},

secondary: null,

target: "self",

type: "Ghost",

zMove: {effect: 'redirect'},

contestType: "Tough",

},

guardianofalola: {

num: 698,

accuracy: true,

basePower: 0,

damageCallback(pokemon, target) {

const hp75 = Math.floor(target.getUndynamaxedHP() \* 3 / 4);

if (

target.volatiles['protect'] || target.volatiles['banefulbunker'] || target.volatiles['kingsshield'] ||

target.volatiles['spikyshield'] || target.side.getSideCondition('matblock')

) {

this.add('-zbroken', target);

return this.clampIntRange(Math.ceil(hp75 / 4 - 0.5), 1);

}

return this.clampIntRange(hp75, 1);

},

category: "Special",

desc: "Deals damage to the target equal to 3/4 of its current HP, rounded down, but not less than 1 HP.",

shortDesc: "Does damage equal to 3/4 target's current HP.",

isNonstandard: "Past",

name: "Guardian of Alola",

pp: 1,

priority: 0,

flags: {},

isZ: "tapuniumz",

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Tough",

},

guardsplit: {

num: 470,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and the target have their Defense and Special Defense stats set to be equal to the average of the user and the target's Defense and Special Defense stats, respectively, rounded down. Stat stage changes are unaffected.",

shortDesc: "Averages Defense and Sp. Def stats with target.",

name: "Guard Split",

pp: 10,

priority: 0,

flags: {protect: 1, mystery: 1},

onHit(target, source) {

const newdef = Math.floor((target.storedStats.def + source.storedStats.def) / 2);

target.storedStats.def = newdef;

source.storedStats.def = newdef;

const newspd = Math.floor((target.storedStats.spd + source.storedStats.spd) / 2);

target.storedStats.spd = newspd;

source.storedStats.spd = newspd;

this.add('-activate', source, 'move: Guard Split', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

guardswap: {

num: 385,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps its Defense and Special Defense stat stage changes with the target.",

shortDesc: "Swaps Defense and Sp. Def changes with target.",

name: "Guard Swap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, authentic: 1, mystery: 1},

onHit(target, source) {

const targetBoosts: SparseBoostsTable = {};

const sourceBoosts: SparseBoostsTable = {};

const defSpd: BoostName[] = ['def', 'spd'];

for (const stat of defSpd) {

targetBoosts[stat] = target.boosts[stat];

sourceBoosts[stat] = source.boosts[stat];

}

source.setBoost(targetBoosts);

target.setBoost(sourceBoosts);

this.add('-swapboost', source, target, 'def, spd', '[from] move: Guard Swap');

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

guillotine: {

num: 12,

accuracy: 30,

basePower: 0,

category: "Physical",

desc: "Deals damage to the target equal to the target's maximum HP. Ignores accuracy and evasiveness modifiers. This attack's accuracy is equal to (user's level - target's level + 30)%, and fails if the target is at a higher level. Pokemon with the Sturdy Ability are immune.",

shortDesc: "OHKOs the target. Fails if user is a lower level.",

name: "Guillotine",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ohko: true,

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 180},

maxMove: {basePower: 130},

contestType: "Cool",

},

gunkshot: {

num: 441,

accuracy: 80,

basePower: 120,

category: "Physical",

desc: "Has a 30% chance to poison the target.",

shortDesc: "30% chance to poison the target.",

name: "Gunk Shot",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Tough",

},

gust: {

num: 16,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Power doubles if the target is using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop.",

shortDesc: "Power doubles during Bounce, Fly, and Sky Drop.",

name: "Gust",

pp: 35,

priority: 0,

flags: {protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Clever",

},

gyroball: {

num: 360,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

let power = Math.floor(25 \* target.getStat('spe') / pokemon.getStat('spe')) + 1;

if (!isFinite(power)) power = 1;

if (power > 150) power = 150;

this.debug(`${power} bp`);

return power;

},

category: "Physical",

desc: "Power is equal to (25 \* target's current Speed / user's current Speed) + 1, rounded down, but not more than 150. If the user's current Speed is 0, this move's power is 1.",

shortDesc: "More power the slower the user than the target.",

name: "Gyro Ball",

pp: 5,

priority: 0,

flags: {bullet: 1, contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cool",

},

hail: {

num: 258,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the weather becomes Hail. At the end of each turn except the last, all active Pokemon lose 1/16 of their maximum HP, rounded down, unless they are an Ice type or have the Ice Body, Magic Guard, Overcoat, or Snow Cloak Abilities. Lasts for 8 turns if the user is holding Icy Rock. Fails if the current weather is Hail.",

shortDesc: "For 5 turns, hail crashes down.",

name: "Hail",

pp: 10,

priority: 0,

flags: {},

weather: 'hail',

secondary: null,

target: "all",

type: "Ice",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

hammerarm: {

num: 359,

accuracy: 90,

basePower: 100,

category: "Physical",

desc: "Lowers the user's Speed by 1 stage.",

shortDesc: "Lowers the user's Speed by 1.",

name: "Hammer Arm",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

self: {

boosts: {

spe: -1,

},

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

happyhour: {

num: 603,

accuracy: true,

basePower: 0,

category: "Status",

shortDesc: "No competitive use.",

name: "Happy Hour",

pp: 30,

priority: 0,

flags: {},

onTryHit(target, source) {

this.add('-activate', target, 'move: Happy Hour');

},

secondary: null,

target: "allySide",

type: "Normal",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Cute",

},

harden: {

num: 106,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 1 stage.",

shortDesc: "Raises the user's Defense by 1.",

name: "Harden",

pp: 30,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 1,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

haze: {

num: 114,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Resets the stat stages of all active Pokemon to 0.",

shortDesc: "Eliminates all stat changes.",

name: "Haze",

pp: 30,

priority: 0,

flags: {authentic: 1},

onHitField() {

this.add('-clearallboost');

for (const pokemon of this.getAllActive()) {

pokemon.clearBoosts();

}

},

secondary: null,

target: "all",

type: "Ice",

zMove: {effect: 'heal'},

contestType: "Beautiful",

},

headbutt: {

num: 29,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Headbutt",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Normal",

contestType: "Tough",

},

headcharge: {

num: 543,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 1/4 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/4 recoil.",

name: "Head Charge",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [1, 4],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

headsmash: {

num: 457,

accuracy: 80,

basePower: 150,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 1/2 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/2 recoil.",

name: "Head Smash",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [1, 2],

secondary: null,

target: "normal",

type: "Rock",

contestType: "Tough",

},

healbell: {

num: 215,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Every Pokemon in the user's party is cured of its major status condition. Active Pokemon with the Soundproof Ability are not cured, unless they are the user.",

shortDesc: "Cures the user's party of all status conditions.",

name: "Heal Bell",

pp: 5,

priority: 0,

flags: {snatch: 1, sound: 1, distance: 1, authentic: 1},

onHit(pokemon, source) {

this.add('-activate', source, 'move: Heal Bell');

const side = pokemon.side;

let success = false;

for (const ally of side.pokemon) {

if (ally !== source && ally.hasAbility('soundproof')) continue;

if (ally.cureStatus()) success = true;

}

return success;

},

target: "allyTeam",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Beautiful",

},

healblock: {

num: 377,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "For 5 turns, the target is prevented from restoring any HP as long as it remains active. During the effect, healing and draining moves are unusable, and Abilities and items that grant healing will not heal the user. If an affected Pokemon uses Baton Pass, the replacement will remain unable to restore its HP. Pain Split and the Regenerator Ability are unaffected.",

shortDesc: "For 5 turns, the foe(s) is prevented from healing.",

isNonstandard: "Past",

name: "Heal Block",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'healblock',

condition: {

duration: 5,

durationCallback(target, source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onStart(pokemon) {

this.add('-start', pokemon, 'move: Heal Block');

},

onDisableMove(pokemon) {

for (const moveSlot of pokemon.moveSlots) {

if (this.dex.getMove(moveSlot.id).flags['heal']) {

pokemon.disableMove(moveSlot.id);

}

}

},

onBeforeMovePriority: 6,

onBeforeMove(pokemon, target, move) {

if (move.flags['heal'] && !move.isZ && !move.isMax) {

this.add('cant', pokemon, 'move: Heal Block', move);

return false;

}

},

onResidualOrder: 17,

onEnd(pokemon) {

this.add('-end', pokemon, 'move: Heal Block');

},

onTryHeal(damage, target, source, effect) {

if ((effect?.id === 'zpower') || this.effectData.isZ) return damage;

return false;

},

},

secondary: null,

target: "allAdjacentFoes",

type: "Psychic",

zMove: {boost: {spa: 2}},

contestType: "Clever",

},

healingwish: {

num: 361,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user faints and the next injured or statused Pokemon brought in has its HP fully restored along with having any major status condition cured. The healing happens before hazards take effect. Is not consumed if the Pokemon sent out is not injured or statused. Fails if the user is the last unfainted Pokemon in its party.",

shortDesc: "User faints. Next hurt Pokemon is fully healed.",

name: "Healing Wish",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

onTryHit(pokemon, target, move) {

if (!this.canSwitch(pokemon.side)) {

delete move.selfdestruct;

return false;

}

},

selfdestruct: "ifHit",

slotCondition: 'healingwish',

condition: {

onSwap(target) {

if (!target.fainted && (target.hp < target.maxhp || target.status)) {

target.heal(target.maxhp);

target.setStatus('');

this.add('-heal', target, target.getHealth, '[from] move: Healing Wish');

target.side.removeSlotCondition(target, 'healingwish');

}

},

},

secondary: null,

target: "self",

type: "Psychic",

contestType: "Beautiful",

},

healorder: {

num: 456,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Heals the user by 50% of its max HP.",

isNonstandard: "Past",

name: "Heal Order",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

secondary: null,

target: "self",

type: "Bug",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

healpulse: {

num: 505,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target restores 1/2 of its maximum HP, rounded half up. If the user has the Mega Launcher Ability, the target instead restores 3/4 of its maximum HP, rounded half down.",

shortDesc: "Heals the target by 50% of its max HP.",

name: "Heal Pulse",

pp: 10,

priority: 0,

flags: {protect: 1, pulse: 1, reflectable: 1, distance: 1, heal: 1, mystery: 1},

onHit(target, source) {

let success = false;

if (source.hasAbility('megalauncher')) {

success = !!this.heal(this.modify(target.baseMaxhp, 0.75));

} else {

success = !!this.heal(Math.ceil(target.baseMaxhp \* 0.5));

}

if (success && target.side !== source.side) {

target.staleness = 'external';

}

return success;

},

secondary: null,

target: "any",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

heartstamp: {

num: 531,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

isNonstandard: "Past",

name: "Heart Stamp",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Psychic",

contestType: "Cute",

},

heartswap: {

num: 391,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps all its stat stage changes with the target.",

shortDesc: "Swaps all stat changes with target.",

isNonstandard: "Past",

name: "Heart Swap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, authentic: 1, mystery: 1},

onHit(target, source) {

const targetBoosts: SparseBoostsTable = {};

const sourceBoosts: SparseBoostsTable = {};

let i: BoostName;

for (i in target.boosts) {

targetBoosts[i] = target.boosts[i];

sourceBoosts[i] = source.boosts[i];

}

target.setBoost(sourceBoosts);

source.setBoost(targetBoosts);

this.add('-swapboost', source, target, '[from] move: Heart Swap');

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {effect: 'crit2'},

contestType: "Clever",

},

heatcrash: {

num: 535,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const targetWeight = target.getWeight();

const pokemonWeight = pokemon.getWeight();

if (pokemonWeight > targetWeight \* 5) {

return 120;

}

if (pokemonWeight > targetWeight \* 4) {

return 100;

}

if (pokemonWeight > targetWeight \* 3) {

return 80;

}

if (pokemonWeight > targetWeight \* 2) {

return 60;

}

return 40;

},

category: "Physical",

desc: "The power of this move depends on (user's weight / target's weight), rounded down. Power is equal to 120 if the result is 5 or more, 100 if 4, 80 if 3, 60 if 2, and 40 if 1 or less. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "More power the heavier the user than the target.",

name: "Heat Crash",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

onTryHit(target, pokemon, move) {

if (target.volatiles['dynamax']) {

this.add('-fail', pokemon, 'Dynamax');

this.attrLastMove('[still]');

return null;

}

},

secondary: null,

target: "normal",

type: "Fire",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Tough",

},

heatwave: {

num: 257,

accuracy: 90,

basePower: 95,

category: "Special",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the foe(s).",

name: "Heat Wave",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "allAdjacentFoes",

type: "Fire",

contestType: "Beautiful",

},

heavyslam: {

num: 484,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const targetWeight = target.getWeight();

const pokemonWeight = pokemon.getWeight();

if (pokemonWeight > targetWeight \* 5) {

return 120;

}

if (pokemonWeight > targetWeight \* 4) {

return 100;

}

if (pokemonWeight > targetWeight \* 3) {

return 80;

}

if (pokemonWeight > targetWeight \* 2) {

return 60;

}

return 40;

},

category: "Physical",

desc: "The power of this move depends on (user's weight / target's weight), rounded down. Power is equal to 120 if the result is 5 or more, 100 if 4, 80 if 3, 60 if 2, and 40 if 1 or less. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "More power the heavier the user than the target.",

name: "Heavy Slam",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

onTryHit(target, pokemon, move) {

if (target.volatiles['dynamax']) {

this.add('-fail', pokemon, 'Dynamax');

this.attrLastMove('[still]');

return null;

}

},

secondary: null,

target: "normal",

type: "Steel",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Tough",

},

helpinghand: {

num: 270,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The power of the target's attack this turn is multiplied by 1.5 (this effect is stackable). Fails if there is no ally adjacent to the user or if the ally already moved this turn, but does not fail if the ally is using a two-turn move.",

shortDesc: "One adjacent ally's move power is 1.5x this turn.",

name: "Helping Hand",

pp: 20,

priority: 5,

flags: {authentic: 1},

volatileStatus: 'helpinghand',

onTryHit(target) {

if (!target.newlySwitched && !this.queue.willMove(target)) return false;

},

condition: {

duration: 1,

onStart(target, source) {

this.effectData.multiplier = 1.5;

this.add('-singleturn', target, 'Helping Hand', '[of] ' + source);

},

onRestart(target, source) {

this.effectData.multiplier \*= 1.5;

this.add('-singleturn', target, 'Helping Hand', '[of] ' + source);

},

onBasePowerPriority: 10,

onBasePower(basePower) {

this.debug('Boosting from Helping Hand: ' + this.effectData.multiplier);

return this.chainModify(this.effectData.multiplier);

},

},

secondary: null,

target: "adjacentAlly",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

hex: {

num: 506,

accuracy: 100,

basePower: 65,

basePowerCallback(pokemon, target, move) {

if (target.status || target.hasAbility('comatose')) return move.basePower \* 2;

return move.basePower;

},

category: "Special",

desc: "Power doubles if the target has a major status condition.",

shortDesc: "Power doubles if the target has a status ailment.",

name: "Hex",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ghost",

zMove: {basePower: 160},

contestType: "Clever",

},

hiddenpower: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "This move's type depends on the user's individual values (IVs), and can be any type but Fairy and Normal.",

shortDesc: "Varies in type based on the user's IVs.",

isNonstandard: "Past",

name: "Hidden Power",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyType(move, pokemon) {

move.type = pokemon.hpType || 'Dark';

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Clever",

},

hiddenpowerbug: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Bug",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Clever",

},

hiddenpowerdark: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Dark",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

hiddenpowerdragon: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Dragon",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Clever",

},

hiddenpowerelectric: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Electric",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Electric",

contestType: "Clever",

},

hiddenpowerfighting: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Fighting",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Clever",

},

hiddenpowerfire: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Fire",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Clever",

},

hiddenpowerflying: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Flying",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Flying",

contestType: "Clever",

},

hiddenpowerghost: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Ghost",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Clever",

},

hiddenpowergrass: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Grass",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Clever",

},

hiddenpowerground: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Ground",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ground",

contestType: "Clever",

},

hiddenpowerice: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Ice",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Clever",

},

hiddenpowerpoison: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Poison",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Poison",

contestType: "Clever",

},

hiddenpowerpsychic: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Psychic",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

hiddenpowerrock: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Rock",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Clever",

},

hiddenpowersteel: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Steel",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

contestType: "Clever",

},

hiddenpowerwater: {

num: 237,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "",

shortDesc: "",

realMove: "Hidden Power",

isNonstandard: "Past",

name: "Hidden Power Water",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

contestType: "Clever",

},

highhorsepower: {

num: 667,

accuracy: 95,

basePower: 95,

category: "Physical",

shortDesc: "No additional effect.",

name: "High Horsepower",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ground",

contestType: "Tough",

},

highjumpkick: {

num: 136,

accuracy: 90,

basePower: 130,

category: "Physical",

desc: "If this attack is not successful, the user loses half of its maximum HP, rounded down, as crash damage. Pokemon with the Magic Guard Ability are unaffected by crash damage.",

shortDesc: "User is hurt by 50% of its max HP if it misses.",

name: "High Jump Kick",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, gravity: 1},

hasCrashDamage: true,

onMoveFail(target, source, move) {

this.damage(source.baseMaxhp / 2, source, source, this.dex.getEffect('High Jump Kick'));

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

holdback: {

num: 610,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "Leaves the target with at least 1 HP.",

shortDesc: "Always leaves the target with at least 1 HP.",

name: "Hold Back",

pp: 40,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

noFaint: true,

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

holdhands: {

num: 615,

accuracy: true,

basePower: 0,

category: "Status",

desc: "No competitive use. Fails if there is no ally adjacent to the user.",

shortDesc: "No competitive use.",

name: "Hold Hands",

pp: 40,

priority: 0,

flags: {authentic: 1},

secondary: null,

target: "adjacentAlly",

type: "Normal",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Cute",

},

honeclaws: {

num: 468,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack and accuracy by 1 stage.",

shortDesc: "Raises the user's Attack and accuracy by 1.",

name: "Hone Claws",

pp: 15,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

accuracy: 1,

},

secondary: null,

target: "self",

type: "Dark",

zMove: {boost: {atk: 1}},

contestType: "Cute",

},

hornattack: {

num: 30,

accuracy: 100,

basePower: 65,

category: "Physical",

shortDesc: "No additional effect.",

name: "Horn Attack",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

horndrill: {

num: 32,

accuracy: 30,

basePower: 0,

category: "Physical",

desc: "Deals damage to the target equal to the target's maximum HP. Ignores accuracy and evasiveness modifiers. This attack's accuracy is equal to (user's level - target's level + 30)%, and fails if the target is at a higher level. Pokemon with the Sturdy Ability are immune.",

shortDesc: "OHKOs the target. Fails if user is a lower level.",

name: "Horn Drill",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ohko: true,

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 180},

maxMove: {basePower: 130},

contestType: "Cool",

},

hornleech: {

num: 532,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Horn Leech",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Grass",

contestType: "Tough",

},

howl: {

num: 336,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the Attack of the user and all allies 1 stage.",

shortDesc: "Raises the user's and ally's Attack by 1.",

name: "Howl",

pp: 40,

priority: 0,

flags: {snatch: 1, sound: 1},

boosts: {

atk: 1,

},

secondary: null,

target: "allies",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Cool",

},

hurricane: {

num: 542,

accuracy: 70,

basePower: 110,

category: "Special",

desc: "Has a 30% chance to confuse the target. This move can hit a target using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop. If the weather is Primordial Sea or Rain Dance, this move does not check accuracy. If the weather is Desolate Land or Sunny Day, this move's accuracy is 50%. If this move is used against a Pokemon holding Utility Umbrella, this move's accuracy remains at 70%.",

shortDesc: "30% chance to confuse target. Can't miss in rain.",

name: "Hurricane",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, distance: 1},

onModifyMove(move, pokemon, target) {

switch (target.effectiveWeather()) {

case 'raindance':

case 'primordialsea':

move.accuracy = true;

break;

case 'sunnyday':

case 'desolateland':

move.accuracy = 50;

break;

}

},

secondary: {

chance: 30,

volatileStatus: 'confusion',

},

target: "any",

type: "Flying",

contestType: "Tough",

},

hydrocannon: {

num: 308,

accuracy: 90,

basePower: 150,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Hydro Cannon",

pp: 5,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

hydropump: {

num: 56,

accuracy: 80,

basePower: 110,

category: "Special",

shortDesc: "No additional effect.",

name: "Hydro Pump",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

hydrovortex: {

num: 642,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Hydro Vortex",

pp: 1,

priority: 0,

flags: {},

isZ: "wateriumz",

secondary: null,

target: "normal",

type: "Water",

contestType: "Cool",

},

hyperbeam: {

num: 63,

accuracy: 90,

basePower: 150,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Hyper Beam",

pp: 5,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

hyperfang: {

num: 158,

accuracy: 90,

basePower: 80,

category: "Physical",

desc: "Has a 10% chance to flinch the target.",

shortDesc: "10% chance to flinch the target.",

isNonstandard: "Past",

name: "Hyper Fang",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'flinch',

},

target: "normal",

type: "Normal",

contestType: "Cool",

},

hyperspacefury: {

num: 621,

accuracy: true,

basePower: 100,

category: "Physical",

desc: "Lowers the user's Defense by 1 stage. This move cannot be used successfully unless the user's current form, while considering Transform, is Hoopa Unbound. If this move is successful, it breaks through the target's Baneful Bunker, Detect, King's Shield, Protect, or Spiky Shield for this turn, allowing other Pokemon to attack the target normally. If the target's side is protected by Crafty Shield, Mat Block, Quick Guard, or Wide Guard, that protection is also broken for this turn and other Pokemon may attack the target's side normally.",

shortDesc: "Hoopa-U: Lowers user's Def by 1; breaks protect.",

isNonstandard: "Past",

name: "Hyperspace Fury",

pp: 5,

priority: 0,

flags: {mirror: 1, authentic: 1},

breaksProtect: true,

onTry(pokemon) {

if (pokemon.species.name === 'Hoopa-Unbound') {

return;

}

this.hint("Only a Pokemon whose form is Hoopa Unbound can use this move.");

if (pokemon.species.name === 'Hoopa') {

this.add('-fail', pokemon, 'move: Hyperspace Fury', '[forme]');

return null;

}

this.add('-fail', pokemon, 'move: Hyperspace Fury');

return null;

},

self: {

boosts: {

def: -1,

},

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Tough",

},

hyperspacehole: {

num: 593,

accuracy: true,

basePower: 80,

category: "Special",

desc: "If this move is successful, it breaks through the target's Baneful Bunker, Detect, King's Shield, Protect, or Spiky Shield for this turn, allowing other Pokemon to attack the target normally. If the target's side is protected by Crafty Shield, Mat Block, Quick Guard, or Wide Guard, that protection is also broken for this turn and other Pokemon may attack the target's side normally.",

shortDesc: "Breaks the target's protection for this turn.",

isNonstandard: "Past",

name: "Hyperspace Hole",

pp: 5,

priority: 0,

flags: {mirror: 1, authentic: 1},

breaksProtect: true,

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

hypervoice: {

num: 304,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent foes.",

name: "Hyper Voice",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

contestType: "Cool",

},

hypnosis: {

num: 95,

accuracy: 60,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

name: "Hypnosis",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

iceball: {

num: 301,

accuracy: 90,

basePower: 30,

basePowerCallback(pokemon, target, move) {

let bp = move.basePower;

if (pokemon.volatiles.iceball && pokemon.volatiles.iceball.hitCount) {

bp \*= Math.pow(2, pokemon.volatiles.iceball.hitCount);

}

if (pokemon.status !== 'slp') pokemon.addVolatile('iceball');

if (pokemon.volatiles.defensecurl) {

bp \*= 2;

}

this.debug("Ice Ball bp: " + bp);

return bp;

},

category: "Physical",

desc: "If this move is successful, the user is locked into this move and cannot make another move until it misses, 5 turns have passed, or the attack cannot be used. Power doubles with each successful hit of this move and doubles again if Defense Curl was used previously by the user. If this move is called by Sleep Talk, the move is used for one turn. If this move hits an active Disguise during the effect, the power multiplier is paused but the turn counter is not, potentially allowing the multiplier to be used on the user's next move after this effect ends.",

shortDesc: "Power doubles with each hit. Repeats for 5 turns.",

isNonstandard: "Past",

name: "Ice Ball",

pp: 20,

priority: 0,

flags: {bullet: 1, contact: 1, protect: 1, mirror: 1},

condition: {

duration: 2,

onLockMove: 'iceball',

onStart() {

this.effectData.hitCount = 1;

},

onRestart() {

this.effectData.hitCount++;

if (this.effectData.hitCount < 5) {

this.effectData.duration = 2;

}

},

onResidual(target) {

if (target.lastMove && target.lastMove.id === 'struggle') {

// don't lock

delete target.volatiles['iceball'];

}

},

},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

icebeam: {

num: 58,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to freeze the target.",

shortDesc: "10% chance to freeze the target.",

name: "Ice Beam",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'frz',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

iceburn: {

num: 554,

accuracy: 90,

basePower: 140,

category: "Special",

desc: "Has a 30% chance to burn the target. This attack charges on the first turn and executes on the second. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Charges turn 1. Hits turn 2. 30% burn.",

name: "Ice Burn",

pp: 5,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

secondary: {

chance: 30,

status: 'brn',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

icefang: {

num: 423,

accuracy: 95,

basePower: 65,

category: "Physical",

desc: "Has a 10% chance to freeze the target and a 10% chance to flinch it.",

shortDesc: "10% chance to freeze. 10% chance to flinch.",

name: "Ice Fang",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondaries: [

{

chance: 10,

status: 'frz',

}, {

chance: 10,

volatileStatus: 'flinch',

},

],

target: "normal",

type: "Ice",

contestType: "Cool",

},

icehammer: {

num: 665,

accuracy: 90,

basePower: 100,

category: "Physical",

desc: "Lowers the user's Speed by 1 stage.",

shortDesc: "Lowers the user's Speed by 1.",

isNonstandard: "Past",

name: "Ice Hammer",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

self: {

boosts: {

spe: -1,

},

},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Tough",

},

icepunch: {

num: 8,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "Has a 10% chance to freeze the target.",

shortDesc: "10% chance to freeze the target.",

name: "Ice Punch",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 10,

status: 'frz',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

iceshard: {

num: 420,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Ice Shard",

pp: 30,

priority: 1,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

iciclecrash: {

num: 556,

accuracy: 90,

basePower: 85,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Icicle Crash",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Ice",

contestType: "Beautiful",

},

iciclespear: {

num: 333,

accuracy: 100,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Icicle Spear",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Ice",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Beautiful",

},

icywind: {

num: 196,

accuracy: 95,

basePower: 55,

category: "Special",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Speed by 1.",

name: "Icy Wind",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "allAdjacentFoes",

type: "Ice",

contestType: "Beautiful",

},

imprison: {

num: 286,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user prevents all opposing Pokemon from using any moves that the user also knows as long as the user remains active.",

shortDesc: "No foe can use any move known by the user.",

name: "Imprison",

pp: 10,

priority: 0,

flags: {snatch: 1, authentic: 1},

volatileStatus: 'imprison',

condition: {

noCopy: true,

onStart(target) {

this.add('-start', target, 'move: Imprison');

},

onFoeDisableMove(pokemon) {

for (const moveSlot of this.effectData.source.moveSlots) {

if (moveSlot.id === 'struggle') continue;

pokemon.disableMove(moveSlot.id, 'hidden');

}

pokemon.maybeDisabled = true;

},

onFoeBeforeMovePriority: 4,

onFoeBeforeMove(attacker, defender, move) {

if (move.id !== 'struggle' && this.effectData.source.hasMove(move.id) && !move.isZ && !move.isMax) {

this.add('cant', attacker, 'move: Imprison', move);

return false;

}

},

},

secondary: null,

pressureTarget: "foeSide",

target: "self",

type: "Psychic",

zMove: {boost: {spd: 2}},

contestType: "Clever",

},

incinerate: {

num: 510,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "The target loses its held item if it is a Berry or a Gem. This move cannot cause Pokemon with the Sticky Hold Ability to lose their held item. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "Destroys the foe(s) Berry/Gem.",

name: "Incinerate",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit(pokemon, source) {

const item = pokemon.getItem();

if ((item.isBerry || item.isGem) && pokemon.takeItem(source)) {

this.add('-enditem', pokemon, item.name, '[from] move: Incinerate');

}

},

secondary: null,

target: "allAdjacentFoes",

type: "Fire",

contestType: "Tough",

},

inferno: {

num: 517,

accuracy: 50,

basePower: 100,

category: "Special",

desc: "Has a 100% chance to burn the target.",

shortDesc: "100% chance to burn the target.",

name: "Inferno",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

infernooverdrive: {

num: 640,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Inferno Overdrive",

pp: 1,

priority: 0,

flags: {},

isZ: "firiumz",

secondary: null,

target: "normal",

type: "Fire",

contestType: "Cool",

},

infestation: {

num: 611,

accuracy: 100,

basePower: 20,

category: "Special",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Infestation",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cute",

},

ingrain: {

num: 275,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user has 1/16 of its maximum HP restored at the end of each turn, but it is prevented from switching out and other Pokemon cannot force the user to switch out. The user can still switch out if it uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the user leaves the field using Baton Pass, the replacement will remain trapped and still receive the healing effect. During the effect, the user can be hit normally by Ground-type attacks and be affected by Spikes, Toxic Spikes, and Sticky Web, even if the user is a Flying type or has the Levitate Ability.",

shortDesc: "Traps/grounds user; heals 1/16 max HP per turn.",

name: "Ingrain",

pp: 20,

priority: 0,

flags: {snatch: 1, nonsky: 1},

volatileStatus: 'ingrain',

condition: {

onStart(pokemon) {

this.add('-start', pokemon, 'move: Ingrain');

},

onResidualOrder: 7,

onResidual(pokemon) {

this.heal(pokemon.baseMaxhp / 16);

},

onTrapPokemon(pokemon) {

pokemon.tryTrap();

},

// groundedness implemented in battle.engine.js:BattlePokemon#isGrounded

onDragOut(pokemon) {

this.add('-activate', pokemon, 'move: Ingrain');

return null;

},

},

secondary: null,

target: "self",

type: "Grass",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

instruct: {

num: 689,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target immediately uses its last used move. Fails if the target has not made a move, if the move has 0 PP, if the target is preparing to use Beak Blast, Focus Punch, or Shell Trap, or if the move is Assist, Beak Blast, Belch, Bide, Celebrate, Copycat, Dynamax Cannon, Focus Punch, Ice Ball, Instruct, King's Shield, Me First, Metronome, Mimic, Mirror Move, Nature Power, Outrage, Petal Dance, Rollout, Shell Trap, Sketch, Sleep Talk, Struggle, Thrash, Transform, Uproar, any two-turn move, any recharge move, or any Z-Move.",

shortDesc: "The target immediately uses its last used move.",

name: "Instruct",

pp: 15,

priority: 0,

flags: {protect: 1, authentic: 1, mystery: 1},

onHit(target, source) {

if (!target.lastMove || target.volatiles['dynamax']) return false;

const lastMove = target.lastMove;

const moveIndex = target.moves.indexOf(lastMove.id);

const noInstruct = [

'assist', 'beakblast', 'bide', 'celebrate', 'copycat', 'dynamaxcannon', 'focuspunch', 'iceball', 'instruct', 'kingsshield', 'mefirst', 'metronome', 'mimic', 'mirrormove', 'naturepower', 'outrage', 'petaldance', 'rollout', 'shelltrap', 'sketch', 'sleeptalk', 'thrash', 'transform',

];

if (

noInstruct.includes(lastMove.id) || lastMove.isZ || lastMove.isMax ||

lastMove.flags['charge'] || lastMove.flags['recharge'] ||

target.volatiles['beakblast'] || target.volatiles['focuspunch'] || target.volatiles['shelltrap'] ||

(target.moveSlots[moveIndex] && target.moveSlots[moveIndex].pp <= 0)

) {

return false;

}

this.add('-singleturn', target, 'move: Instruct', '[of] ' + source);

this.runMove(target.lastMove.id, target, target.lastMoveTargetLoc!);

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

iondeluge: {

num: 569,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Causes Normal-type moves to become Electric type this turn. The effect happens after other effects that change a move's type.",

shortDesc: "Normal moves become Electric type this turn.",

isNonstandard: "Past",

name: "Ion Deluge",

pp: 25,

priority: 1,

flags: {},

pseudoWeather: 'iondeluge',

condition: {

duration: 1,

onStart(target) {

this.add('-fieldactivate', 'move: Ion Deluge');

},

onModifyTypePriority: -2,

onModifyType(move) {

if (move.type === 'Normal') {

move.type = 'Electric';

this.debug(move.name + "'s type changed to Electric");

}

},

},

secondary: null,

target: "all",

type: "Electric",

zMove: {boost: {spa: 1}},

contestType: "Beautiful",

},

irondefense: {

num: 334,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 2 stages.",

shortDesc: "Raises the user's Defense by 2.",

name: "Iron Defense",

pp: 15,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 2,

},

secondary: null,

target: "self",

type: "Steel",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

ironhead: {

num: 442,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Iron Head",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Steel",

contestType: "Tough",

},

irontail: {

num: 231,

accuracy: 75,

basePower: 100,

category: "Physical",

desc: "Has a 30% chance to lower the target's Defense by 1 stage.",

shortDesc: "30% chance to lower the target's Defense by 1.",

name: "Iron Tail",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

boosts: {

def: -1,

},

},

target: "normal",

type: "Steel",

contestType: "Cool",

},

jawlock: {

num: 746,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Prevents the user and the target from switching out. The user and the target can still switch out if either of them is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if either the user or the target leaves the field.",

shortDesc: "Prevents both user and target from switching out.",

name: "Jaw Lock",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

onHit(target, source, move) {

source.addVolatile('trapped', target, move, 'trapper');

target.addVolatile('trapped', source, move, 'trapper');

},

secondary: null,

target: "normal",

type: "Dark",

},

judgment: {

num: 449,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "This move's type depends on the user's held Plate.",

shortDesc: "Type varies based on the held Plate.",

isNonstandard: "Past",

name: "Judgment",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyType(move, pokemon) {

if (pokemon.ignoringItem()) return;

const item = pokemon.getItem();

if (item.id && item.onPlate && !item.zMove) {

move.type = item.onPlate;

}

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

jumpkick: {

num: 26,

accuracy: 95,

basePower: 100,

category: "Physical",

desc: "If this attack is not successful, the user loses half of its maximum HP, rounded down, as crash damage. Pokemon with the Magic Guard Ability are unaffected by crash damage.",

shortDesc: "User is hurt by 50% of its max HP if it misses.",

isNonstandard: "Past",

name: "Jump Kick",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, gravity: 1},

hasCrashDamage: true,

onMoveFail(target, source, move) {

this.damage(source.baseMaxhp / 2, source, source, this.dex.getEffect('Jump Kick'));

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

junglehealing: {

num: 816,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Each Pokemon on the user's side restores 1/4 of its maximum HP, rounded half up, and has its status condition cured.",

shortDesc: "User and allies: healed 1/4 max HP, status cured.",

name: "Jungle Healing",

pp: 10,

priority: 0,

flags: {distance: 1, heal: 1, authentic: 1, mystery: 1},

onHit(pokemon) {

pokemon.cureStatus();

},

heal: [1, 4],

secondary: null,

target: "allies",

type: "Grass",

},

karatechop: {

num: 2,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

isNonstandard: "Past",

name: "Karate Chop",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

kinesis: {

num: 134,

accuracy: 80,

basePower: 0,

category: "Status",

desc: "Lowers the target's accuracy by 1 stage.",

shortDesc: "Lowers the target's accuracy by 1.",

name: "Kinesis",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

accuracy: -1,

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {evasion: 1}},

contestType: "Clever",

},

kingsshield: {

num: 588,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn, and Pokemon trying to make contact with the user have their Attack lowered by 1 stage. Non-damaging moves go through this protection. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Protects from damaging attacks. Contact: -1 Atk.",

name: "King's Shield",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'kingsshield',

onTryHit(pokemon) {

return !!this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'Protect');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect'] || move.category === 'Status') {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Protect');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

if (move.flags['contact']) {

this.boost({atk: -1}, source, target, this.dex.getActiveMove("King's Shield"));

}

return this.NOT\_FAIL;

},

onHit(target, source, move) {

if (move.isZOrMaxPowered && move.flags['contact']) {

this.boost({atk: -1}, source, target, this.dex.getActiveMove("King's Shield"));

}

},

},

secondary: null,

target: "self",

type: "Steel",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cool",

},

knockoff: {

num: 282,

accuracy: 100,

basePower: 65,

category: "Physical",

desc: "If the target is holding an item that can be removed from it, ignoring the Sticky Hold Ability, this move's power is multiplied by 1.5. If the user has not fainted, the target loses its held item. This move cannot cause Pokemon with the Sticky Hold Ability to lose their held item or cause a Kyogre, a Groudon, a Giratina, an Arceus, a Genesect, a Silvally, a Zacian, or a Zamazenta to lose their Blue Orb, Red Orb, Griseous Orb, Plate, Drive, Memory, Rusted Sword, or Rusted Shield respectively. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "1.5x damage if foe holds an item. Removes item.",

name: "Knock Off",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onBasePower(basePower, source, target, move) {

const item = target.getItem();

if (!this.singleEvent('TakeItem', item, target.itemData, target, target, move, item)) return;

if (item.id) {

return this.chainModify(1.5);

}

},

onAfterHit(target, source) {

if (source.hp) {

const item = target.takeItem();

if (item) {

this.add('-enditem', target, item.name, '[from] move: Knock Off', '[of] ' + source);

}

}

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

landswrath: {

num: 616,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent foes.",

isNonstandard: "Past",

name: "Land's Wrath",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Ground",

zMove: {basePower: 185},

contestType: "Beautiful",

},

laserfocus: {

num: 673,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the next turn, the user's attacks will be critical hits.",

shortDesc: "Until the end of the next turn, user's moves crit.",

name: "Laser Focus",

pp: 30,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'laserfocus',

condition: {

duration: 2,

onStart(pokemon, source, effect) {

if (effect && (['imposter', 'psychup', 'transform'].includes(effect.id))) {

this.add('-start', pokemon, 'move: Laser Focus', '[silent]');

} else {

this.add('-start', pokemon, 'move: Laser Focus');

}

},

onRestart(pokemon) {

this.effectData.duration = 2;

this.add('-start', pokemon, 'move: Laser Focus');

},

onModifyCritRatio(critRatio) {

return 5;

},

onEnd(pokemon) {

this.add('-end', pokemon, 'move: Laser Focus', '[silent]');

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Cool",

},

lashout: {

num: 808,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "Power doubles if the user had a stat stage lowered this turn.",

shortDesc: "2x power if the user had a stat lowered this turn.",

name: "Lash Out",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onBasePower(basePower, source) {

if (source.statsLoweredThisTurn) {

this.debug('lashout buff');

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Dark",

},

lastresort: {

num: 387,

accuracy: 100,

basePower: 140,

category: "Physical",

desc: "This move fails unless the user knows this move and at least one other move, and has used all the other moves it knows at least once each since it became active or Transformed.",

shortDesc: "Fails unless each known move has been used.",

name: "Last Resort",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onTry(pokemon, target) {

if (pokemon.moveSlots.length < 2) return false; // Last Resort fails unless the user knows at least 2 moves

let hasLastResort = false; // User must actually have Last Resort for it to succeed

for (const moveSlot of pokemon.moveSlots) {

if (moveSlot.id === 'lastresort') {

hasLastResort = true;

continue;

}

if (!moveSlot.used) return false;

}

return hasLastResort;

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

lavaplume: {

num: 436,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 30% chance to burn the target.",

shortDesc: "30% chance to burn adjacent Pokemon.",

name: "Lava Plume",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'brn',

},

target: "allAdjacent",

type: "Fire",

contestType: "Tough",

},

leafage: {

num: 670,

accuracy: 100,

basePower: 40,

category: "Physical",

shortDesc: "No additional effect.",

name: "Leafage",

pp: 40,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Tough",

},

leafblade: {

num: 348,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Leaf Blade",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

leafstorm: {

num: 437,

accuracy: 90,

basePower: 130,

category: "Special",

desc: "Lowers the user's Special Attack by 2 stages.",

shortDesc: "Lowers the user's Sp. Atk by 2.",

name: "Leaf Storm",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

self: {

boosts: {

spa: -2,

},

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Beautiful",

},

leaftornado: {

num: 536,

accuracy: 90,

basePower: 65,

category: "Special",

desc: "Has a 50% chance to lower the target's accuracy by 1 stage.",

shortDesc: "50% chance to lower the target's accuracy by 1.",

name: "Leaf Tornado",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Grass",

contestType: "Cool",

},

leechlife: {

num: 141,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Leech Life",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Bug",

contestType: "Clever",

},

leechseed: {

num: 73,

accuracy: 90,

basePower: 0,

category: "Status",

desc: "The Pokemon at the user's position steals 1/8 of the target's maximum HP, rounded down, at the end of each turn. If Big Root is held by the recipient, the HP recovered is 1.3x normal, rounded half down. If the target uses Baton Pass, the replacement will continue being leeched. If the target switches out or uses Rapid Spin successfully, the effect ends. Grass-type Pokemon are immune to this move on use, but not its effect.",

shortDesc: "1/8 of target's HP is restored to user every turn.",

name: "Leech Seed",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'leechseed',

condition: {

onStart(target) {

this.add('-start', target, 'move: Leech Seed');

},

onResidualOrder: 8,

onResidual(pokemon) {

const target = this.effectData.source.side.active[pokemon.volatiles['leechseed'].sourcePosition];

if (!target || target.fainted || target.hp <= 0) {

this.debug('Nothing to leech into');

return;

}

const damage = this.damage(pokemon.baseMaxhp / 8, pokemon, target);

if (damage) {

this.heal(damage, target, pokemon);

}

},

},

onTryImmunity(target) {

return !target.hasType('Grass');

},

secondary: null,

target: "normal",

type: "Grass",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

leer: {

num: 43,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Defense by 1 stage.",

shortDesc: "Lowers the foe(s) Defense by 1.",

name: "Leer",

pp: 30,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

def: -1,

},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Cool",

},

letssnuggleforever: {

num: 726,

accuracy: true,

basePower: 190,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Let's Snuggle Forever",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "mimikiumz",

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Cool",

},

lick: {

num: 122,

accuracy: 100,

basePower: 30,

category: "Physical",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze the target.",

name: "Lick",

pp: 30,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Ghost",

contestType: "Cute",

},

lifedew: {

num: 791,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Each Pokemon on the user's side restores 1/4 of its maximum HP, rounded half up.",

shortDesc: "Heals the user and its allies by 1/4 their max HP.",

name: "Life Dew",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1, authentic: 1},

heal: [1, 4],

secondary: null,

target: "allies",

type: "Water",

},

lightofruin: {

num: 617,

accuracy: 90,

basePower: 140,

category: "Special",

desc: "If the target lost HP, the user takes recoil damage equal to 1/2 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/2 recoil.",

isNonstandard: "Past",

name: "Light of Ruin",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

recoil: [1, 2],

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Beautiful",

},

lightscreen: {

num: 113,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members take 0.5x damage from special attacks, or 0.66x damage if in a Double Battle. Damage is not reduced further with Aurora Veil. Critical hits ignore this effect. It is removed from the user's side if the user or an ally is successfully hit by Brick Break, Psychic Fangs, or Defog. Lasts for 8 turns if the user is holding Light Clay. Fails if the effect is already active on the user's side.",

shortDesc: "For 5 turns, special damage to allies is halved.",

name: "Light Screen",

pp: 30,

priority: 0,

flags: {snatch: 1},

sideCondition: 'lightscreen',

condition: {

duration: 5,

durationCallback(target, source, effect) {

if (source?.hasItem('lightclay')) {

return 8;

}

return 5;

},

onAnyModifyDamage(damage, source, target, move) {

if (target !== source && target.side === this.effectData.target && this.getCategory(move) === 'Special') {

if (!target.getMoveHitData(move).crit && !move.infiltrates) {

this.debug('Light Screen weaken');

if (target.side.active.length > 1) return this.chainModify([0xAAC, 0x1000]);

return this.chainModify(0.5);

}

}

},

onStart(side) {

this.add('-sidestart', side, 'move: Light Screen');

},

onResidualOrder: 21,

onResidualSubOrder: 1,

onEnd(side) {

this.add('-sideend', side, 'move: Light Screen');

},

},

secondary: null,

target: "allySide",

type: "Psychic",

zMove: {boost: {spd: 1}},

contestType: "Beautiful",

},

lightthatburnsthesky: {

num: 723,

accuracy: true,

basePower: 200,

category: "Special",

desc: "This move becomes a physical attack if the user's Attack is greater than its Special Attack, including stat stage changes. This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Physical if user's Atk > Sp. Atk. Ignores Abilities.",

isNonstandard: "Past",

name: "Light That Burns the Sky",

pp: 1,

priority: 0,

flags: {},

onModifyMove(move, pokemon) {

if (pokemon.getStat('atk', false, true) > pokemon.getStat('spa', false, true)) move.category = 'Physical';

},

ignoreAbility: true,

isZ: "ultranecroziumz",

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

liquidation: {

num: 710,

accuracy: 100,

basePower: 85,

category: "Physical",

desc: "Has a 20% chance to lower the target's Defense by 1 stage.",

shortDesc: "20% chance to lower the target's Defense by 1.",

name: "Liquidation",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

boosts: {

def: -1,

},

},

target: "normal",

type: "Water",

contestType: "Cool",

},

lockon: {

num: 199,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the next turn, the target cannot avoid the user's moves, even if the target is in the middle of a two-turn move. The effect ends if either the user or the target leaves the field. Fails if this effect is active for the user.",

shortDesc: "User's next move will not miss the target.",

name: "Lock-On",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onTryHit(target, source) {

if (source.volatiles['lockon']) return false;

},

onHit(target, source) {

source.addVolatile('lockon', target);

this.add('-activate', source, 'move: Lock-On', '[of] ' + target);

},

condition: {

noCopy: true, // doesn't get copied by Baton Pass

duration: 2,

onSourceInvulnerabilityPriority: 1,

onSourceInvulnerability(target, source, move) {

if (move && source === this.effectData.target && target === this.effectData.source) return 0;

},

onSourceAccuracy(accuracy, target, source, move) {

if (move && source === this.effectData.target && target === this.effectData.source) return true;

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

lovelykiss: {

num: 142,

accuracy: 75,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

isNonstandard: "Past",

name: "Lovely Kiss",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

lowkick: {

num: 67,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const targetWeight = target.getWeight();

if (targetWeight >= 2000) {

return 120;

}

if (targetWeight >= 1000) {

return 100;

}

if (targetWeight >= 500) {

return 80;

}

if (targetWeight >= 250) {

return 60;

}

if (targetWeight >= 100) {

return 40;

}

return 20;

},

category: "Physical",

desc: "This move's power is 20 if the target weighs less than 10 kg, 40 if less than 25 kg, 60 if less than 50 kg, 80 if less than 100 kg, 100 if less than 200 kg, and 120 if greater than or equal to 200 kg.",

shortDesc: "More power the heavier the target.",

name: "Low Kick",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onTryHit(target, pokemon, move) {

if (target.volatiles['dynamax']) {

this.add('-fail', pokemon, 'Dynamax');

this.attrLastMove('[still]');

return null;

}

},

secondary: null,

target: "normal",

type: "Fighting",

zMove: {basePower: 160},

contestType: "Tough",

},

lowsweep: {

num: 490,

accuracy: 100,

basePower: 65,

category: "Physical",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the target's Speed by 1.",

name: "Low Sweep",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Fighting",

contestType: "Clever",

},

luckychant: {

num: 381,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members cannot be struck by a critical hit. Fails if the effect is already active on the user's side.",

shortDesc: "For 5 turns, shields user's party from critical hits.",

isNonstandard: "Past",

name: "Lucky Chant",

pp: 30,

priority: 0,

flags: {snatch: 1},

sideCondition: 'luckychant',

condition: {

duration: 5,

onStart(side) {

this.add('-sidestart', side, 'move: Lucky Chant'); // "The Lucky Chant shielded [side.name]'s team from critical hits!"

},

onCriticalHit: false,

onResidualOrder: 21,

onResidualSubOrder: 5,

onEnd(side) {

this.add('-sideend', side, 'move: Lucky Chant'); // "[side.name]'s team's Lucky Chant wore off!"

},

},

secondary: null,

target: "allySide",

type: "Normal",

zMove: {boost: {evasion: 1}},

contestType: "Cute",

},

lunardance: {

num: 461,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user faints and the Pokemon brought out to replace it has its HP and PP fully restored along with having any major status condition cured. The new Pokemon is sent out at the end of the turn, and the healing happens before hazards take effect. Fails if the user is the last unfainted Pokemon in its party.",

shortDesc: "User faints. Replacement is fully healed, with PP.",

isNonstandard: "Past",

name: "Lunar Dance",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1, dance: 1},

onTryHit(pokemon, target, move) {

if (!this.canSwitch(pokemon.side)) {

delete move.selfdestruct;

return false;

}

},

selfdestruct: "ifHit",

sideCondition: 'lunardance',

condition: {

duration: 2,

onStart(side, source) {

this.debug('Lunar Dance started on ' + side.name);

this.effectData.positions = [];

for (const i of side.active.keys()) {

this.effectData.positions[i] = false;

}

this.effectData.positions[source.position] = true;

},

onRestart(side, source) {

this.effectData.positions[source.position] = true;

},

onSwitchInPriority: 1,

onSwitchIn(target) {

const positions: boolean[] = this.effectData.positions;

if (target.position !== this.effectData.sourcePosition) {

return;

}

if (!target.fainted) {

target.heal(target.maxhp);

target.setStatus('');

for (const moveSlot of target.moveSlots) {

moveSlot.pp = moveSlot.maxpp;

}

this.add('-heal', target, target.getHealth, '[from] move: Lunar Dance');

positions[target.position] = false;

}

if (!positions.some(affected => affected === true)) {

target.side.removeSideCondition('lunardance');

}

},

},

secondary: null,

target: "self",

type: "Psychic",

contestType: "Beautiful",

},

lunge: {

num: 679,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 100% chance to lower the target's Attack by 1 stage.",

shortDesc: "100% chance to lower the target's Attack by 1.",

name: "Lunge",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

atk: -1,

},

},

target: "normal",

type: "Bug",

contestType: "Cute",

},

lusterpurge: {

num: 295,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "Has a 50% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "50% chance to lower the target's Sp. Def by 1.",

isNonstandard: "Past",

name: "Luster Purge",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Psychic",

contestType: "Clever",

},

machpunch: {

num: 183,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Mach Punch",

pp: 30,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

magicalleaf: {

num: 345,

accuracy: true,

basePower: 60,

category: "Special",

shortDesc: "This move does not check accuracy.",

name: "Magical Leaf",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Beautiful",

},

magiccoat: {

num: 277,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the turn, the user is unaffected by certain non-damaging moves directed at it and will instead use such moves against the original user. Moves reflected in this way are unable to be reflected again by this or the Magic Bounce Ability's effect. Spikes, Stealth Rock, Sticky Web, and Toxic Spikes can only be reflected once per side, by the leftmost Pokemon under this or the Magic Bounce Ability's effect. The Lightning Rod and Storm Drain Abilities redirect their respective moves before this move takes effect.",

shortDesc: "Bounces back certain non-damaging moves.",

name: "Magic Coat",

pp: 15,

priority: 4,

flags: {},

volatileStatus: 'magiccoat',

condition: {

duration: 1,

onStart(target, source, effect) {

this.add('-singleturn', target, 'move: Magic Coat');

if (effect?.effectType === 'Move') {

this.effectData.pranksterBoosted = effect.pranksterBoosted;

}

},

onTryHitPriority: 2,

onTryHit(target, source, move) {

if (target === source || move.hasBounced || !move.flags['reflectable']) {

return;

}

const newMove = this.dex.getActiveMove(move.id);

newMove.hasBounced = true;

newMove.pranksterBoosted = this.effectData.pranksterBoosted;

this.useMove(newMove, target, source);

return null;

},

onAllyTryHitSide(target, source, move) {

if (target.side === source.side || move.hasBounced || !move.flags['reflectable']) {

return;

}

const newMove = this.dex.getActiveMove(move.id);

newMove.hasBounced = true;

newMove.pranksterBoosted = false;

this.useMove(newMove, this.effectData.target, source);

return null;

},

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {boost: {spd: 2}},

contestType: "Beautiful",

},

magicpowder: {

num: 750,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to become a Psychic type. Fails if the target is an Arceus or a Silvally, or if the target is already purely Psychic type.",

shortDesc: "Changes the target's type to Psychic.",

name: "Magic Powder",

pp: 20,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target) {

if (target.getTypes().join() === 'Psychic' || !target.setType('Psychic')) return false;

this.add('-start', target, 'typechange', 'Psychic');

},

secondary: null,

target: "normal",

type: "Psychic",

},

magicroom: {

num: 478,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the held items of all active Pokemon have no effect. An item's effect of causing forme changes is unaffected, but any other effects from such items are negated. During the effect, Fling and Natural Gift are prevented from being used by all active Pokemon. If this move is used during the effect, the effect ends.",

shortDesc: "For 5 turns, all held items have no effect.",

name: "Magic Room",

pp: 10,

priority: 0,

flags: {mirror: 1},

pseudoWeather: 'magicroom',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onStart(target, source) {

this.add('-fieldstart', 'move: Magic Room', '[of] ' + source);

},

onRestart(target, source) {

this.field.removePseudoWeather('magicroom');

},

// Item suppression implemented in Pokemon.ignoringItem() within sim/pokemon.js

onResidualOrder: 25,

onEnd() {

this.add('-fieldend', 'move: Magic Room', '[of] ' + this.effectData.source);

},

},

secondary: null,

target: "all",

type: "Psychic",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

magmastorm: {

num: 463,

accuracy: 75,

basePower: 100,

category: "Special",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

isNonstandard: "Past",

name: "Magma Storm",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Fire",

contestType: "Tough",

},

magnetbomb: {

num: 443,

accuracy: true,

basePower: 60,

category: "Physical",

shortDesc: "This move does not check accuracy.",

isNonstandard: "Past",

name: "Magnet Bomb",

pp: 20,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

contestType: "Cool",

},

magneticflux: {

num: 602,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the Defense and Special Defense of Pokemon on the user's side with the Plus or Minus Abilities by 1 stage.",

shortDesc: "Raises Def, Sp. Def of allies with Plus/Minus by 1.",

name: "Magnetic Flux",

pp: 20,

priority: 0,

flags: {snatch: 1, distance: 1, authentic: 1},

onHitSide(side, source, move) {

const targets = [];

for (const pokemon of side.active) {

if (pokemon.hasAbility(['plus', 'minus'])) {

targets.push(pokemon);

}

}

if (!targets.length) return false;

let didSomething = false;

for (const target of targets) {

didSomething = this.boost({def: 1, spd: 1}, target, source, move, false, true) || didSomething;

}

return didSomething;

},

secondary: null,

target: "allySide",

type: "Electric",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

magnetrise: {

num: 393,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user is immune to Ground-type attacks and the effects of Spikes, Toxic Spikes, Sticky Web, and the Arena Trap Ability as long as it remains active. If the user uses Baton Pass, the replacement will gain the effect. Ingrain, Smack Down, Thousand Arrows, and Iron Ball override this move if the user is under any of their effects. Fails if the user is already under this effect or the effects of Ingrain, Smack Down, or Thousand Arrows.",

shortDesc: "For 5 turns, the user has immunity to Ground.",

name: "Magnet Rise",

pp: 10,

priority: 0,

flags: {snatch: 1, gravity: 1},

volatileStatus: 'magnetrise',

condition: {

duration: 5,

onStart(target) {

if (target.volatiles['smackdown'] || target.volatiles['ingrain']) return false;

this.add('-start', target, 'Magnet Rise');

},

onImmunity(type) {

if (type === 'Ground') return false;

},

onResidualOrder: 15,

onEnd(target) {

this.add('-end', target, 'Magnet Rise');

},

},

secondary: null,

target: "self",

type: "Electric",

zMove: {boost: {evasion: 1}},

contestType: "Clever",

},

magnitude: {

num: 222,

accuracy: 100,

basePower: 0,

category: "Physical",

desc: "The power of this move varies; 5% chances for 10 and 150 power, 10% chances for 30 and 110 power, 20% chances for 50 and 90 power, and 30% chance for 70 power. Damage doubles if the target is using Dig.",

shortDesc: "Hits adjacent Pokemon. Power varies; 2x on Dig.",

isNonstandard: "Past",

name: "Magnitude",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onModifyMove(move, pokemon) {

const i = this.random(100);

if (i < 5) {

move.magnitude = 4;

move.basePower = 10;

} else if (i < 15) {

move.magnitude = 5;

move.basePower = 30;

} else if (i < 35) {

move.magnitude = 6;

move.basePower = 50;

} else if (i < 65) {

move.magnitude = 7;

move.basePower = 70;

} else if (i < 85) {

move.magnitude = 8;

move.basePower = 90;

} else if (i < 95) {

move.magnitude = 9;

move.basePower = 110;

} else {

move.magnitude = 10;

move.basePower = 150;

}

},

onUseMoveMessage(pokemon, target, move) {

this.add('-activate', pokemon, 'move: Magnitude', move.magnitude);

},

secondary: null,

target: "allAdjacent",

type: "Ground",

zMove: {basePower: 140},

maxMove: {basePower: 140},

contestType: "Tough",

},

maliciousmoonsault: {

num: 696,

accuracy: true,

basePower: 180,

category: "Physical",

desc: "Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "Damage doubles if the target used Minimize.",

isNonstandard: "Past",

name: "Malicious Moonsault",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "inciniumz",

secondary: null,

target: "normal",

type: "Dark",

contestType: "Cool",

},

matblock: {

num: 561,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and its party members are protected from damaging attacks made by other Pokemon, including allies, during this turn. Fails unless it is the user's first turn on the field, if the user moves last this turn, or if this move is already in effect for the user's side.",

shortDesc: "Protects allies from damaging attacks. Turn 1 only.",

name: "Mat Block",

pp: 10,

priority: 0,

flags: {snatch: 1, nonsky: 1},

stallingMove: true,

sideCondition: 'matblock',

onTryHitSide(side, source) {

if (source.activeMoveActions > 1) {

this.hint("Mat Block only works on your first turn out.");

return false;

}

},

condition: {

duration: 1,

onStart(target, source) {

this.add('-singleturn', source, 'Mat Block');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect']) {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move && (move.target === 'self' || move.category === 'Status')) return;

this.add('-activate', target, 'move: Mat Block', move.name);

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

return this.NOT\_FAIL;

},

},

secondary: null,

target: "allySide",

type: "Fighting",

zMove: {boost: {def: 1}},

contestType: "Cool",

},

maxairstream: {

num: 766,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Speed of each Pokemon on the user's side is raised by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: +1 Speed.",

name: "Max Airstream",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.active) {

this.boost({spe: 1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Flying",

contestType: "Cool",

},

maxdarkness: {

num: 772,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Special Defense of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 Sp. Def.",

name: "Max Darkness",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.foe.active) {

this.boost({spd: -1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Dark",

contestType: "Cool",

},

maxflare: {

num: 757,

accuracy: true,

basePower: 100,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Sunny Day begins.",

shortDesc: "Base move affects power. Starts Sunny Day.",

name: "Max Flare",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setWeather('sunnyday');

},

},

target: "adjacentFoe",

type: "Fire",

contestType: "Cool",

},

maxflutterby: {

num: 758,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Special Attack of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 Sp. Atk.",

name: "Max Flutterby",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.foe.active) {

this.boost({spa: -1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Bug",

contestType: "Cool",

},

maxgeyser: {

num: 765,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Rain Dance begins.",

shortDesc: "Base move affects power. Starts Rain Dance.",

name: "Max Geyser",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setWeather('raindance');

},

},

target: "adjacentFoe",

type: "Water",

contestType: "Cool",

},

maxguard: {

num: 743,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from nearly all attacks made by other Pokemon during this turn, including Max and G-Max Moves. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Max Guard, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Protects user from moves & Max Moves this turn.",

name: "Max Guard",

pp: 5,

priority: 4,

flags: {},

isMax: true,

stallingMove: true,

volatileStatus: 'maxguard',

onPrepareHit(pokemon) {

return !!this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'Max Guard');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (move.isMax && move.breaksProtect) return;

/\*\* moves blocked by Max Guard but not Protect \*/

const overrideBypassProtect = [

'block', 'flowershield', 'gearup', 'magneticflux', 'phantomforce', 'psychup', 'teatime', 'transform',

];

const blockedByMaxGuard = (this.dex.getMove(move.id).flags['protect'] ||

move.isZ || move.isMax || overrideBypassProtect.includes(move.id));

if (!blockedByMaxGuard) {

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Max Guard');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

return this.NOT\_FAIL;

},

},

secondary: null,

target: "self",

type: "Normal",

contestType: "Cool",

},

maxhailstorm: {

num: 763,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Hail begins.",

shortDesc: "Base move affects power. Starts Hail.",

name: "Max Hailstorm",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setWeather('hail');

},

},

target: "adjacentFoe",

type: "Ice",

contestType: "Cool",

},

maxknuckle: {

num: 761,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Boosts the user and its allies' Attack by 1 stage. BP scales with the base move's BP.",

shortDesc: "Base move affects power. Allies: +1 Attack.",

name: "Max Knuckle",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.active) {

this.boost({atk: 1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Fighting",

contestType: "Cool",

},

maxlightning: {

num: 759,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Electric Terrain begins.",

shortDesc: "Base move affects power. Starts Electric Terrain.",

name: "Max Lightning",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setTerrain('electricterrain');

},

},

target: "adjacentFoe",

type: "Electric",

contestType: "Cool",

},

maxmindstorm: {

num: 769,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Psychic Terrain begins.",

shortDesc: "Base move affects power. Starts Psychic Terrain.",

name: "Max Mindstorm",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setTerrain('psychicterrain');

},

},

target: "adjacentFoe",

type: "Psychic",

contestType: "Cool",

},

maxooze: {

num: 764,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Special Attack of each Pokemon on the user's side is raised by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: +1 Sp. Atk.",

name: "Max Ooze",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.active) {

this.boost({spa: 1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Poison",

contestType: "Cool",

},

maxovergrowth: {

num: 773,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Grassy Terrain begins.",

shortDesc: "Base move affects power. Starts Grassy Terrain.",

name: "Max Overgrowth",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setTerrain('grassyterrain');

},

},

target: "adjacentFoe",

type: "Grass",

contestType: "Cool",

},

maxphantasm: {

num: 762,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Defense of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 Defense.",

name: "Max Phantasm",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.foe.active) {

this.boost({def: -1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Ghost",

contestType: "Cool",

},

maxquake: {

num: 771,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Special Defense of each Pokemon on the user's side is raised by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: +1 Sp. Def.",

name: "Max Quake",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.active) {

this.boost({spd: 1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Ground",

contestType: "Cool",

},

maxrockfall: {

num: 770,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Sandstorm begins.",

shortDesc: "Base move affects power. Starts Sandstorm.",

name: "Max Rockfall",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setWeather('sandstorm');

},

},

target: "adjacentFoe",

type: "Rock",

contestType: "Cool",

},

maxstarfall: {

num: 767,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the effect of Misty Terrain begins.",

shortDesc: "Base move affects power. Starts Misty Terrain.",

name: "Max Starfall",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

this.field.setTerrain('mistyterrain');

},

},

target: "adjacentFoe",

type: "Fairy",

contestType: "Cool",

},

maxsteelspike: {

num: 774,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Defense of each Pokemon on the user's side is raised by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Allies: +1 Defense.",

name: "Max Steelspike",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.active) {

this.boost({def: 1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Steel",

contestType: "Cool",

},

maxstrike: {

num: 760,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Speed of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 Speed.",

name: "Max Strike",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.foe.active) {

this.boost({spe: -1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Normal",

contestType: "Cool",

},

maxwyrmwind: {

num: 768,

accuracy: true,

basePower: 10,

category: "Physical",

desc: "Power is equal to the base move's Max Move power. If this move is successful, the Attack of each Pokemon on the opposing side is lowered by 1 stage, even if they have a substitute.",

shortDesc: "Base move affects power. Foes: -1 Attack.",

name: "Max Wyrmwind",

pp: 5,

priority: 0,

flags: {},

isMax: true,

self: {

onHit(source) {

if (!source.volatiles['dynamax']) return;

for (const pokemon of source.side.foe.active) {

this.boost({atk: -1}, pokemon);

}

},

},

target: "adjacentFoe",

type: "Dragon",

contestType: "Cool",

},

meanlook: {

num: 212,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Prevents the target from switching out.",

name: "Mean Look",

pp: 5,

priority: 0,

flags: {reflectable: 1, mirror: 1},

onHit(target, source, move) {

return target.addVolatile('trapped', source, move, 'trapper');

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Beautiful",

},

meditate: {

num: 96,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack by 1 stage.",

shortDesc: "Raises the user's Attack by 1.",

isNonstandard: "Past",

name: "Meditate",

pp: 40,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {boost: {atk: 1}},

contestType: "Beautiful",

},

mefirst: {

num: 382,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user uses the move the target chose for use this turn against it, if possible, with its power multiplied by 1.5. The move must be a damaging move other than Beak Blast, Chatter, Counter, Covet, Focus Punch, Me First, Metal Burst, Mirror Coat, Shell Trap, Struggle, Thief, or any Z-Move. Fails if the target moves before the user. Ignores the target's substitute for the purpose of copying the move.",

shortDesc: "Copies a foe at 1.5x power. User must be faster.",

isNonstandard: "Past",

name: "Me First",

pp: 20,

priority: 0,

flags: {protect: 1, authentic: 1},

onTryHit(target, pokemon) {

const action = this.queue.willMove(target);

if (!action) return false;

const noMeFirst = [

'beakblast', 'chatter', 'counter', 'covet', 'focuspunch', 'mefirst', 'metalburst', 'mirrorcoat', 'shelltrap', 'struggle', 'thief',

];

const move = this.dex.getActiveMove(action.move.id);

if (action.zmove || move.isZ || move.isMax) return false;

if (move.category === 'Status' || noMeFirst.includes(move.id)) return false;

pokemon.addVolatile('mefirst');

this.useMove(move, pokemon, target);

return null;

},

condition: {

duration: 1,

onBasePowerPriority: 12,

onBasePower(basePower) {

return this.chainModify(1.5);

},

},

secondary: null,

target: "adjacentFoe",

type: "Normal",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

megadrain: {

num: 72,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Mega Drain",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "normal",

type: "Grass",

zMove: {basePower: 120},

contestType: "Clever",

},

megahorn: {

num: 224,

accuracy: 85,

basePower: 120,

category: "Physical",

shortDesc: "No additional effect.",

name: "Megahorn",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cool",

},

megakick: {

num: 25,

accuracy: 75,

basePower: 120,

category: "Physical",

shortDesc: "No additional effect.",

name: "Mega Kick",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

megapunch: {

num: 5,

accuracy: 85,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Mega Punch",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

memento: {

num: 262,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack and Special Attack by 2 stages. The user faints unless this move misses or there is no target. Fails entirely if this move hits a substitute, but does not fail if the target's stats cannot be changed.",

shortDesc: "Lowers target's Attack, Sp. Atk by 2. User faints.",

name: "Memento",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

boosts: {

atk: -2,

spa: -2,

},

selfdestruct: "ifHit",

secondary: null,

target: "normal",

type: "Dark",

zMove: {effect: 'healreplacement'},

contestType: "Tough",

},

menacingmoonrazemaelstrom: {

num: 725,

accuracy: true,

basePower: 200,

category: "Special",

desc: "This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Ignores the Abilities of other Pokemon.",

isNonstandard: "Past",

name: "Menacing Moonraze Maelstrom",

pp: 1,

priority: 0,

flags: {},

isZ: "lunaliumz",

ignoreAbility: true,

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

metalburst: {

num: 368,

accuracy: 100,

basePower: 0,

damageCallback(pokemon) {

if (!pokemon.volatiles['metalburst']) return 0;

return pokemon.volatiles['metalburst'].damage || 1;

},

category: "Physical",

desc: "Deals damage to the last opposing Pokemon to hit the user with an attack this turn equal to 1.5 times the HP lost by the user from that attack, rounded down. If the user did not lose HP from the attack, this move deals 1 HP of damage instead. If that opposing Pokemon's position is no longer in use and there is another opposing Pokemon on the field, the damage is done to it instead. Only the last hit of a multi-hit attack is counted. Fails if the user was not hit by an opposing Pokemon's attack this turn.",

shortDesc: "If hit by an attack, returns 1.5x damage.",

name: "Metal Burst",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('metalburst');

},

onTryHit(target, source, move) {

if (!source.volatiles['metalburst']) return false;

if (source.volatiles['metalburst'].position === null) return false;

},

condition: {

duration: 1,

noCopy: true,

onStart(target, source, move) {

this.effectData.position = null;

this.effectData.damage = 0;

},

onRedirectTargetPriority: -1,

onRedirectTarget(target, source, source2) {

if (source !== this.effectData.target) return;

return source.side.foe.active[this.effectData.position];

},

onDamagingHit(damage, target, source, effect) {

if (source.side !== target.side) {

this.effectData.position = source.position;

this.effectData.damage = 1.5 \* damage;

}

},

},

secondary: null,

target: "scripted",

type: "Steel",

contestType: "Cool",

},

metalclaw: {

num: 232,

accuracy: 95,

basePower: 50,

category: "Physical",

desc: "Has a 10% chance to raise the user's Attack by 1 stage.",

shortDesc: "10% chance to raise the user's Attack by 1.",

name: "Metal Claw",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

self: {

boosts: {

atk: 1,

},

},

},

target: "normal",

type: "Steel",

contestType: "Cool",

},

metalsound: {

num: 319,

accuracy: 85,

basePower: 0,

category: "Status",

desc: "Lowers the target's Special Defense by 2 stages.",

shortDesc: "Lowers the target's Sp. Def by 2.",

name: "Metal Sound",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1, mystery: 1},

boosts: {

spd: -2,

},

secondary: null,

target: "normal",

type: "Steel",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

meteorassault: {

num: 794,

accuracy: 100,

basePower: 150,

category: "Physical",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Meteor Assault",

pp: 5,

priority: 0,

flags: {protect: 1, recharge: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Fighting",

},

meteorbeam: {

num: 800,

accuracy: 90,

basePower: 120,

category: "Special",

desc: "This attack charges on the first turn and executes on the second. Raises the user's Special Attack by 1 stage on the first turn. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Raises user's Sp. Atk by 1 on turn 1. Hits turn 2.",

name: "Meteor Beam",

pp: 10,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

this.boost({spa: 1}, attacker, attacker, move);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

secondary: null,

target: "normal",

type: "Rock",

},

meteormash: {

num: 309,

accuracy: 90,

basePower: 90,

category: "Physical",

desc: "Has a 20% chance to raise the user's Attack by 1 stage.",

shortDesc: "20% chance to raise the user's Attack by 1.",

name: "Meteor Mash",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 20,

self: {

boosts: {

atk: 1,

},

},

},

target: "normal",

type: "Steel",

contestType: "Cool",

},

metronome: {

num: 118,

accuracy: true,

basePower: 0,

category: "Status",

desc: "A random move is selected for use, other than After You, Apple Acid, Assist, Aura Wheel, Baneful Bunker, Beak Blast, Behemoth Bash, Behemoth Blade, Belch, Bestow, Body Press, Branch Poke, Breaking Swipe, Celebrate, Chatter, Clangorous Soul, Copycat, Counter, Covet, Crafty Shield, Decorate, Destiny Bond, Detect, Diamond Storm, Double Iron Bash, Dragon Ascent, Drum Beating, Dynamax Cannon, Endure, Eternabeam, False Surrender, Feint, Fleur Cannon, Focus Punch, Follow Me, Freeze Shock, Grav Apple, Helping Hand, Hold Hands, Hyperspace Fury, Hyperspace Hole, Ice Burn, Instruct, King's Shield, Life Dew, Light of Ruin, Mat Block, Me First, Meteor Assault, Metronome, Mimic, Mind Blown, Mirror Coat, Mirror Move, Moongeist Beam, Nature Power, Nature's Madness, Obstruct, Origin Pulse, Overdrive, Photon Geyser, Plasma Fists, Precipice Blades, Protect, Pyro Ball, Quash, Quick Guard, Rage Powder, Relic Song, Secret Sword, Shell Trap, Sketch, Sleep Talk, Snap Trap, Snarl, Snatch, Snore, Spectral Thief, Spiky Shield, Spirit Break, Spotlight, Steam Eruption, Steel Beam, Strange Steam, Struggle, Sunsteel Strike, Switcheroo, Techno Blast, Thief, Thousand Arrows, Thousand Waves, Transform, Trick, V-create, or Wide Guard.",

shortDesc: "Picks a random move.",

name: "Metronome",

pp: 10,

priority: 0,

flags: {},

noMetronome: [

"After You", "Apple Acid", "Assist", "Aura Wheel", "Baneful Bunker", "Beak Blast", "Behemoth Bash", "Behemoth Blade", "Belch", "Bestow", "Body Press", "Branch Poke", "Breaking Swipe", "Celebrate", "Chatter", "Clangorous Soul", "Copycat", "Counter", "Covet", "Crafty Shield", "Decorate", "Destiny Bond", "Detect", "Diamond Storm", "Double Iron Bash", "Dragon Ascent", "Drum Beating", "Dynamax Cannon", "Endure", "Eternabeam", "False Surrender", "Feint", "Fleur Cannon", "Focus Punch", "Follow Me", "Freeze Shock", "Grav Apple", "Helping Hand", "Hold Hands", "Hyperspace Fury", "Hyperspace Hole", "Ice Burn", "Instruct", "Jungle Healing", "King's Shield", "Life Dew", "Light of Ruin", "Mat Block", "Me First", "Meteor Assault", "Metronome", "Mimic", "Mind Blown", "Mirror Coat", "Mirror Move", "Moongeist Beam", "Nature Power", "Nature's Madness", "Obstruct", "Origin Pulse", "Overdrive", "Photon Geyser", "Plasma Fists", "Precipice Blades", "Protect", "Pyro Ball", "Quash", "Quick Guard", "Rage Powder", "Relic Song", "Secret Sword", "Shell Trap", "Sketch", "Sleep Talk", "Snap Trap", "Snarl", "Snatch", "Snore", "Spectral Thief", "Spiky Shield", "Spirit Break", "Spotlight", "Steam Eruption", "Steel Beam", "Strange Steam", "Struggle", "Sunsteel Strike", "Surging Strikes", "Switcheroo", "Techno Blast", "Thief", "Thousand Arrows", "Thousand Waves", "Transform", "Trick", "V-create", "Wicked Blow", "Wide Guard",

],

onHit(target, source, effect) {

const moves: MoveData[] = [];

for (const id in Moves) {

const move = Moves[id];

if (move.realMove) continue;

if (move.isZ || move.isMax || move.isNonstandard) continue;

if (effect.noMetronome!.includes(move.name)) continue;

if (this.dex.getMove(id).gen > this.gen) continue;

moves.push(move);

}

let randomMove = '';

if (moves.length) {

moves.sort((a, b) => a.num! - b.num!);

randomMove = this.sample(moves).name;

}

if (!randomMove) {

return false;

}

this.useMove(randomMove, target);

},

secondary: null,

target: "self",

type: "Normal",

contestType: "Cute",

},

milkdrink: {

num: 208,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Heals the user by 50% of its max HP.",

name: "Milk Drink",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

mimic: {

num: 102,

accuracy: true,

basePower: 0,

category: "Status",

desc: "While the user remains active, this move is replaced by the last move used by the target. The copied move has the maximum PP for that move. Fails if the target has not made a move, if the user has Transformed, if the user already knows the move, or if the move is Chatter, Mimic, Sketch, Struggle, Transform, or any Z-Move.",

shortDesc: "The last move the target used replaces this one.",

name: "Mimic",

pp: 10,

priority: 0,

flags: {protect: 1, authentic: 1, mystery: 1},

onHit(target, source) {

const disallowedMoves = ['chatter', 'mimic', 'sketch', 'struggle', 'transform'];

const move = target.lastMove;

if (source.transformed || !move || disallowedMoves.includes(move.id) || source.moves.includes(move.id)) {

return false;

}

if (move.isZ || move.isMax) return false;

const mimicIndex = source.moves.indexOf('mimic');

if (mimicIndex < 0) return false;

source.moveSlots[mimicIndex] = {

move: move.name,

id: move.id,

pp: move.pp,

maxpp: move.pp,

target: move.target,

disabled: false,

used: false,

virtual: true,

};

this.add('-start', source, 'Mimic', move.name);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {accuracy: 1}},

contestType: "Cute",

},

mindblown: {

num: 720,

accuracy: 100,

basePower: 150,

category: "Special",

desc: "Whether or not this move is successful and even if it would cause fainting, the user loses 1/2 of its maximum HP, rounded up, unless the user has the Magic Guard Ability. This move is prevented from executing and the user does not lose HP if any active Pokemon has the Damp Ability, or if this move is Fire type and the user is affected by Powder or the weather is Primordial Sea.",

shortDesc: "User loses 50% max HP. Hits adjacent Pokemon.",

isNonstandard: "Past",

name: "Mind Blown",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

mindBlownRecoil: true,

onAfterMove(pokemon, target, move) {

if (move.mindBlownRecoil && !move.multihit) {

this.damage(Math.round(pokemon.maxhp / 2), pokemon, pokemon, this.dex.getEffect('Mind Blown'), true);

}

},

secondary: null,

target: "allAdjacent",

type: "Fire",

contestType: "Cool",

},

mindreader: {

num: 170,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the next turn, the target cannot avoid the user's moves, even if the target is in the middle of a two-turn move. The effect ends if either the user or the target leaves the field. Fails if this effect is active for the user.",

shortDesc: "User's next move will not miss the target.",

name: "Mind Reader",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onTryHit(target, source) {

if (source.volatiles['lockon']) return false;

},

onHit(target, source) {

source.addVolatile('lockon', target);

this.add('-activate', source, 'move: Mind Reader', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

minimize: {

num: 107,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's evasiveness by 2 stages. Whether or not the user's evasiveness was changed, Body Slam, Dragon Rush, Flying Press, Heat Crash, Heavy Slam, Malicious Moonsault, Steamroller, and Stomp will not check accuracy and have their damage doubled if used against the user while it is active.",

shortDesc: "Raises the user's evasiveness by 2.",

name: "Minimize",

pp: 10,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'minimize',

condition: {

noCopy: true,

onSourceModifyDamage(damage, source, target, move) {

const boostedMoves = [

'stomp', 'steamroller', 'bodyslam', 'flyingpress', 'dragonrush', 'heatcrash', 'heavyslam', 'maliciousmoonsault',

];

if (boostedMoves.includes(move.id)) {

return this.chainModify(2);

}

},

onAccuracy(accuracy, target, source, move) {

const boostedMoves = [

'stomp', 'steamroller', 'bodyslam', 'flyingpress', 'dragonrush', 'heatcrash', 'heavyslam', 'maliciousmoonsault',

];

if (boostedMoves.includes(move.id)) {

return true;

}

return accuracy;

},

},

boosts: {

evasion: 2,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

miracleeye: {

num: 357,

accuracy: true,

basePower: 0,

category: "Status",

desc: "As long as the target remains active, its evasiveness stat stage is ignored during accuracy checks against it if it is greater than 0, and Psychic-type attacks can hit the target if it is a Dark type. Fails if the target is already affected, or affected by Foresight or Odor Sleuth.",

shortDesc: "Psychic hits Dark. Evasiveness ignored.",

isNonstandard: "Past",

name: "Miracle Eye",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'miracleeye',

onTryHit(target) {

if (target.volatiles['foresight']) return false;

},

condition: {

noCopy: true,

onStart(pokemon) {

this.add('-start', pokemon, 'Miracle Eye');

},

onNegateImmunity(pokemon, type) {

if (pokemon.hasType('Dark') && type === 'Psychic') return false;

},

onModifyBoost(boosts) {

if (boosts.evasion && boosts.evasion > 0) {

boosts.evasion = 0;

}

},

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

mirrorcoat: {

num: 243,

accuracy: 100,

basePower: 0,

damageCallback(pokemon) {

if (!pokemon.volatiles['mirrorcoat']) return 0;

return pokemon.volatiles['mirrorcoat'].damage || 1;

},

category: "Special",

desc: "Deals damage to the last opposing Pokemon to hit the user with a special attack this turn equal to twice the HP lost by the user from that attack. If the user did not lose HP from the attack, this move deals 1 HP of damage instead. If that opposing Pokemon's position is no longer in use and there is another opposing Pokemon on the field, the damage is done to it instead. Only the last hit of a multi-hit attack is counted. Fails if the user was not hit by an opposing Pokemon's special attack this turn.",

shortDesc: "If hit by special attack, returns double damage.",

name: "Mirror Coat",

pp: 20,

priority: -5,

flags: {protect: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('mirrorcoat');

},

onTryHit(target, source, move) {

if (!source.volatiles['mirrorcoat']) return false;

if (source.volatiles['mirrorcoat'].position === null) return false;

},

condition: {

duration: 1,

noCopy: true,

onStart(target, source, move) {

this.effectData.position = null;

this.effectData.damage = 0;

},

onRedirectTargetPriority: -1,

onRedirectTarget(target, source, source2) {

if (source !== this.effectData.target) return;

return source.side.foe.active[this.effectData.position];

},

onDamagingHit(damage, target, source, move) {

if (source.side !== target.side && this.getCategory(move) === 'Special') {

this.effectData.position = source.position;

this.effectData.damage = 2 \* damage;

}

},

},

secondary: null,

target: "scripted",

type: "Psychic",

contestType: "Beautiful",

},

mirrormove: {

num: 119,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user uses the last move used by the target. The copied move is used against that target, if possible. Fails if the target has not made a move, or if the last move used cannot be copied by this move.",

shortDesc: "User uses the target's last used move against it.",

isNonstandard: "Past",

name: "Mirror Move",

pp: 20,

priority: 0,

flags: {},

onTryHit(target, pokemon) {

const move = target.lastMove;

if (!move || !move.flags['mirror'] || move.isZ || move.isMax) {

return false;

}

this.useMove(move.id, pokemon, target);

return null;

},

secondary: null,

target: "normal",

type: "Flying",

zMove: {boost: {atk: 2}},

contestType: "Clever",

},

mirrorshot: {

num: 429,

accuracy: 85,

basePower: 65,

category: "Special",

desc: "Has a 30% chance to lower the target's accuracy by 1 stage.",

shortDesc: "30% chance to lower the target's accuracy by 1.",

isNonstandard: "Past",

name: "Mirror Shot",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Steel",

contestType: "Beautiful",

},

mist: {

num: 54,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members are protected from having their stat stages lowered by other Pokemon. Fails if the effect is already active on the user's side.",

shortDesc: "For 5 turns, protects user's party from stat drops.",

name: "Mist",

pp: 30,

priority: 0,

flags: {snatch: 1},

sideCondition: 'mist',

condition: {

duration: 5,

onBoost(boost, target, source, effect) {

if (effect.effectType === 'Move' && effect.infiltrates && target.side !== source.side) return;

if (source && target !== source) {

let showMsg = false;

let i: BoostName;

for (i in boost) {

if (boost[i]! < 0) {

delete boost[i];

showMsg = true;

}

}

if (showMsg && !(effect as ActiveMove).secondaries) {

this.add('-activate', target, 'move: Mist');

}

}

},

onStart(side) {

this.add('-sidestart', side, 'Mist');

},

onResidualOrder: 21,

onResidualSubOrder: 3,

onEnd(side) {

this.add('-sideend', side, 'Mist');

},

},

secondary: null,

target: "allySide",

type: "Ice",

zMove: {effect: 'heal'},

contestType: "Beautiful",

},

mistball: {

num: 296,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "Has a 50% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "50% chance to lower the target's Sp. Atk by 1.",

isNonstandard: "Past",

name: "Mist Ball",

pp: 5,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

spa: -1,

},

},

target: "normal",

type: "Psychic",

contestType: "Clever",

},

mistyexplosion: {

num: 802,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "If the current terrain is Misty Terrain and the user is grounded, this move's power is multiplied by 1.5. The user faints after using this move, even if this move fails for having no target. This move is prevented from executing if any active Pokemon has the Damp Ability.",

shortDesc: "User faints. User on Misty Terrain: 1.5x power.",

name: "Misty Explosion",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

selfdestruct: "always",

onBasePower(basePower, source) {

if (this.field.isTerrain('mistyterrain') && source.isGrounded()) {

this.debug('misty terrain boost');

return this.chainModify(1.5);

}

},

secondary: null,

target: "allAdjacent",

type: "Fairy",

},

mistyterrain: {

num: 581,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the terrain becomes Misty Terrain. During the effect, the power of Dragon-type attacks used against grounded Pokemon is multiplied by 0.5 and grounded Pokemon cannot be inflicted with a major status condition nor confusion. Camouflage transforms the user into a Fairy type, Nature Power becomes Moonblast, and Secret Power has a 30% chance to lower Special Attack by 1 stage. Fails if the current terrain is Misty Terrain.",

shortDesc: "5 turns. Can't status,-Dragon power vs grounded.",

name: "Misty Terrain",

pp: 10,

priority: 0,

flags: {nonsky: 1},

terrain: 'mistyterrain',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasItem('terrainextender')) {

return 8;

}

return 5;

},

onSetStatus(status, target, source, effect) {

if (!target.isGrounded() || target.isSemiInvulnerable()) return;

if (effect && ((effect as Move).status || effect.id === 'yawn')) {

this.add('-activate', target, 'move: Misty Terrain');

}

return false;

},

onTryAddVolatile(status, target, source, effect) {

if (!target.isGrounded() || target.isSemiInvulnerable()) return;

if (status.id === 'confusion') {

if (effect.effectType === 'Move' && !effect.secondaries) this.add('-activate', target, 'move: Misty Terrain');

return null;

}

},

onBasePowerPriority: 6,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Dragon' && defender.isGrounded() && !defender.isSemiInvulnerable()) {

this.debug('misty terrain weaken');

return this.chainModify(0.5);

}

},

onStart(battle, source, effect) {

if (effect?.effectType === 'Ability') {

this.add('-fieldstart', 'move: Misty Terrain', '[from] ability: ' + effect, '[of] ' + source);

} else {

this.add('-fieldstart', 'move: Misty Terrain');

}

},

onResidualOrder: 21,

onResidualSubOrder: 2,

onEnd(side) {

this.add('-fieldend', 'Misty Terrain');

},

},

secondary: null,

target: "all",

type: "Fairy",

zMove: {boost: {spd: 1}},

contestType: "Beautiful",

},

moonblast: {

num: 585,

accuracy: 100,

basePower: 95,

category: "Special",

desc: "Has a 30% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "30% chance to lower the target's Sp. Atk by 1.",

name: "Moonblast",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

boosts: {

spa: -1,

},

},

target: "normal",

type: "Fairy",

contestType: "Beautiful",

},

moongeistbeam: {

num: 714,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Ignores the Abilities of other Pokemon.",

name: "Moongeist Beam",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

ignoreAbility: true,

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

moonlight: {

num: 236,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP if Delta Stream or no weather conditions are in effect or if the user is holding Utility Umbrella, 2/3 of its maximum HP if the weather is Desolate Land or Sunny Day, and 1/4 of its maximum HP if the weather is Hail, Primordial Sea, Rain Dance, or Sandstorm, all rounded half down.",

shortDesc: "Heals the user by a weather-dependent amount.",

name: "Moonlight",

pp: 5,

priority: 0,

flags: {snatch: 1, heal: 1},

onHit(pokemon) {

let factor = 0.5;

switch (pokemon.effectiveWeather()) {

case 'sunnyday':

case 'desolateland':

factor = 0.667;

break;

case 'raindance':

case 'primordialsea':

case 'sandstorm':

case 'hail':

factor = 0.25;

break;

}

return !!this.heal(this.modify(pokemon.maxhp, factor));

},

secondary: null,

target: "self",

type: "Fairy",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

morningsun: {

num: 234,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP if Delta Stream or no weather conditions are in effect or if the user is holding Utility Umbrella, 2/3 of its maximum HP if the weather is Desolate Land or Sunny Day, and 1/4 of its maximum HP if the weather is Hail, Primordial Sea, Rain Dance, or Sandstorm, all rounded half down.",

shortDesc: "Heals the user by a weather-dependent amount.",

name: "Morning Sun",

pp: 5,

priority: 0,

flags: {snatch: 1, heal: 1},

onHit(pokemon) {

let factor = 0.5;

switch (pokemon.effectiveWeather()) {

case 'sunnyday':

case 'desolateland':

factor = 0.667;

break;

case 'raindance':

case 'primordialsea':

case 'sandstorm':

case 'hail':

factor = 0.25;

break;

}

return !!this.heal(this.modify(pokemon.maxhp, factor));

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

mudbomb: {

num: 426,

accuracy: 85,

basePower: 65,

category: "Special",

desc: "Has a 30% chance to lower the target's accuracy by 1 stage.",

shortDesc: "30% chance to lower the target's accuracy by 1.",

isNonstandard: "Past",

name: "Mud Bomb",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Ground",

contestType: "Cute",

},

mudshot: {

num: 341,

accuracy: 95,

basePower: 55,

category: "Special",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the target's Speed by 1.",

name: "Mud Shot",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Ground",

contestType: "Tough",

},

mudslap: {

num: 189,

accuracy: 100,

basePower: 20,

category: "Special",

desc: "Has a 100% chance to lower the target's accuracy by 1 stage.",

shortDesc: "100% chance to lower the target's accuracy by 1.",

name: "Mud-Slap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Ground",

contestType: "Cute",

},

mudsport: {

num: 300,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, all Electric-type attacks used by any active Pokemon have their power multiplied by 0.33. Fails if this effect is already active.",

shortDesc: "For 5 turns, Electric-type attacks have 1/3 power.",

isNonstandard: "Past",

name: "Mud Sport",

pp: 15,

priority: 0,

flags: {nonsky: 1},

pseudoWeather: 'mudsport',

condition: {

duration: 5,

onStart(side, source) {

this.add('-fieldstart', 'move: Mud Sport', '[of] ' + source);

},

onBasePowerPriority: 1,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Electric') {

this.debug('mud sport weaken');

return this.chainModify([0x548, 0x1000]);

}

},

onResidualOrder: 21,

onEnd() {

this.add('-fieldend', 'move: Mud Sport');

},

},

secondary: null,

target: "all",

type: "Ground",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

muddywater: {

num: 330,

accuracy: 85,

basePower: 90,

category: "Special",

desc: "Has a 30% chance to lower the target's accuracy by 1 stage.",

shortDesc: "30% chance to lower the foe(s) accuracy by 1.",

name: "Muddy Water",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: {

chance: 30,

boosts: {

accuracy: -1,

},

},

target: "allAdjacentFoes",

type: "Water",

contestType: "Tough",

},

multiattack: {

num: 718,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "This move's type depends on the user's held Memory.",

shortDesc: "Type varies based on the held Memory.",

name: "Multi-Attack",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onModifyType(move, pokemon) {

if (pokemon.ignoringItem()) return;

move.type = this.runEvent('Memory', pokemon, null, move, 'Normal');

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 185},

maxMove: {basePower: 95},

contestType: "Tough",

},

mysticalfire: {

num: 595,

accuracy: 100,

basePower: 75,

category: "Special",

desc: "Has a 100% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "100% chance to lower the target's Sp. Atk by 1.",

name: "Mystical Fire",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spa: -1,

},

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

nastyplot: {

num: 417,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Attack by 2 stages.",

shortDesc: "Raises the user's Sp. Atk by 2.",

name: "Nasty Plot",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

spa: 2,

},

secondary: null,

target: "self",

type: "Dark",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

naturalgift: {

num: 363,

accuracy: 100,

basePower: 0,

category: "Physical",

desc: "The type and power of this move depend on the user's held Berry, and the Berry is lost. Fails if the user is not holding a Berry, if the user has the Klutz Ability, or if Embargo or Magic Room is in effect for the user.",

shortDesc: "Power and type depends on the user's Berry.",

isNonstandard: "Past",

name: "Natural Gift",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyType(move, pokemon) {

if (pokemon.ignoringItem()) return;

const item = pokemon.getItem();

if (!item.naturalGift) return;

move.type = item.naturalGift.type;

},

onPrepareHit(target, pokemon, move) {

if (pokemon.ignoringItem()) return false;

const item = pokemon.getItem();

if (!item.naturalGift) return false;

move.basePower = item.naturalGift.basePower;

pokemon.setItem('');

pokemon.lastItem = item.id;

pokemon.usedItemThisTurn = true;

this.runEvent('AfterUseItem', pokemon, null, null, item);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Clever",

},

naturepower: {

num: 267,

accuracy: true,

basePower: 0,

category: "Status",

desc: "This move calls another move for use based on the battle terrain. Tri Attack on the regular Wi-Fi terrain, Thunderbolt during Electric Terrain, Moonblast during Misty Terrain, Energy Ball during Grassy Terrain, and Psychic during Psychic Terrain.",

shortDesc: "Attack depends on terrain (default Tri Attack).",

name: "Nature Power",

pp: 20,

priority: 0,

flags: {},

onTryHit(target, pokemon) {

let move = 'triattack';

if (this.field.isTerrain('electricterrain')) {

move = 'thunderbolt';

} else if (this.field.isTerrain('grassyterrain')) {

move = 'energyball';

} else if (this.field.isTerrain('mistyterrain')) {

move = 'moonblast';

} else if (this.field.isTerrain('psychicterrain')) {

move = 'psychic';

}

this.useMove(move, pokemon, target);

return null;

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

naturesmadness: {

num: 717,

accuracy: 90,

basePower: 0,

damageCallback(pokemon, target) {

return this.clampIntRange(Math.floor(target.getUndynamaxedHP() / 2), 1);

},

category: "Special",

desc: "Deals damage to the target equal to half of its current HP, rounded down, but not less than 1 HP.",

shortDesc: "Does damage equal to 1/2 target's current HP.",

isNonstandard: "Past",

name: "Nature's Madness",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Tough",

},

needlearm: {

num: 302,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

isNonstandard: "Past",

name: "Needle Arm",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Grass",

contestType: "Clever",

},

neverendingnightmare: {

num: 636,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Never-Ending Nightmare",

pp: 1,

priority: 0,

flags: {},

isZ: "ghostiumz",

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

nightdaze: {

num: 539,

accuracy: 95,

basePower: 85,

category: "Special",

desc: "Has a 40% chance to lower the target's accuracy by 1 stage.",

shortDesc: "40% chance to lower the target's accuracy by 1.",

name: "Night Daze",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 40,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Dark",

contestType: "Cool",

},

nightmare: {

num: 171,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to lose 1/4 of its maximum HP, rounded down, at the end of each turn as long as it is asleep. This move does not affect the target unless it is asleep. The effect ends when the target wakes up, even if it falls asleep again in the same turn.",

shortDesc: "A sleeping target is hurt by 1/4 max HP per turn.",

isNonstandard: "Past",

name: "Nightmare",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

volatileStatus: 'nightmare',

condition: {

noCopy: true,

onStart(pokemon) {

if (pokemon.status !== 'slp' && !pokemon.hasAbility('comatose')) {

return false;

}

this.add('-start', pokemon, 'Nightmare');

},

onResidualOrder: 9,

onResidual(pokemon) {

this.damage(pokemon.baseMaxhp / 4);

},

},

secondary: null,

target: "normal",

type: "Ghost",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

nightshade: {

num: 101,

accuracy: 100,

basePower: 0,

damage: 'level',

category: "Special",

desc: "Deals damage to the target equal to the user's level.",

shortDesc: "Does damage equal to the user's level.",

name: "Night Shade",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Clever",

},

nightslash: {

num: 400,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Night Slash",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Dark",

contestType: "Cool",

},

nobleroar: {

num: 568,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack and Special Attack by 1 stage.",

shortDesc: "Lowers the target's Attack and Sp. Atk by 1.",

name: "Noble Roar",

pp: 30,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

boosts: {

atk: -1,

spa: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

noretreat: {

num: 748,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage, but it becomes prevented from switching out. The user can still switch out if it uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the user leaves the field using Baton Pass, the replacement will remain trapped. Fails if the user has already been prevented from switching by this effect.",

shortDesc: "Raises all stats by 1 (not acc/eva). Traps user.",

name: "No Retreat",

pp: 5,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'noretreat',

onTryHit(target, source, move) {

if (target.volatiles['noretreat']) return false;

if (target.volatiles['trapped']) {

delete move.volatileStatus;

}

},

condition: {

onStart(pokemon) {

this.add('-start', pokemon, 'move: No Retreat');

},

onTrapPokemon(pokemon) {

pokemon.tryTrap();

},

},

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

secondary: null,

target: "self",

type: "Fighting",

},

nuzzle: {

num: 609,

accuracy: 100,

basePower: 20,

category: "Physical",

desc: "Has a 100% chance to paralyze the target.",

shortDesc: "100% chance to paralyze the target.",

name: "Nuzzle",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cute",

},

oblivionwing: {

num: 613,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "The user recovers 3/4 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 75% of the damage dealt.",

isNonstandard: "Past",

name: "Oblivion Wing",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, distance: 1, heal: 1},

drain: [3, 4],

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

obstruct: {

num: 792,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn, and Pokemon trying to make contact with the user have their Defense lowered by 2 stages. Non-damaging moves go through this protection. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Max Guard, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Protects from damaging attacks. Contact: -2 Def.",

name: "Obstruct",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'obstruct',

onTryHit(pokemon) {

return !!this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'Protect');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect'] || move.category === 'Status') {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Protect');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

if (move.flags['contact']) {

this.boost({def: -2}, source, target, this.dex.getActiveMove("Obstruct"));

}

return this.NOT\_FAIL;

},

onHit(target, source, move) {

if (move.isZOrMaxPowered && move.flags['contact']) {

this.boost({def: -2}, source, target, this.dex.getActiveMove("Obstruct"));

}

},

},

secondary: null,

target: "self",

type: "Dark",

},

oceanicoperetta: {

num: 697,

accuracy: true,

basePower: 195,

category: "Special",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Oceanic Operetta",

pp: 1,

priority: 0,

flags: {},

isZ: "primariumz",

secondary: null,

target: "normal",

type: "Water",

contestType: "Cool",

},

octazooka: {

num: 190,

accuracy: 85,

basePower: 65,

category: "Special",

desc: "Has a 50% chance to lower the target's accuracy by 1 stage.",

shortDesc: "50% chance to lower the target's accuracy by 1.",

name: "Octazooka",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

accuracy: -1,

},

},

target: "normal",

type: "Water",

contestType: "Tough",

},

octolock: {

num: 753,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Prevents the target from switching out. At the end of each turn during effect, the target's Defense and Special Defense are lowered by 1 stage. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Traps target, lowers Def and SpD by 1 each turn.",

name: "Octolock",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onTryImmunity(target) {

return this.dex.getImmunity('trapped', target);

},

volatileStatus: 'octolock',

condition: {

onStart(pokemon, source) {

this.add('-start', pokemon, 'move: Octolock', '[of] ' + source);

},

onResidualOrder: 11,

onResidual(pokemon) {

const source = this.effectData.source;

if (source && (!source.isActive || source.hp <= 0 || !source.activeTurns)) {

delete pokemon.volatiles['octolock'];

this.add('-end', pokemon, 'Octolock', '[partiallytrapped]', '[silent]');

return;

}

this.boost({def: -1, spd: -1}, pokemon, source, this.dex.getActiveMove('octolock'));

},

onTrapPokemon(pokemon) {

if (this.effectData.source && this.effectData.source.isActive) pokemon.tryTrap();

},

},

secondary: null,

target: "normal",

type: "Fighting",

},

odorsleuth: {

num: 316,

accuracy: true,

basePower: 0,

category: "Status",

desc: "As long as the target remains active, its evasiveness stat stage is ignored during accuracy checks against it if it is greater than 0, and Normal- and Fighting-type attacks can hit the target if it is a Ghost type. Fails if the target is already affected, or affected by Foresight or Miracle Eye.",

shortDesc: "Fighting, Normal hit Ghost. Evasiveness ignored.",

isNonstandard: "Past",

name: "Odor Sleuth",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1, mystery: 1},

volatileStatus: 'foresight',

onTryHit(target) {

if (target.volatiles['miracleeye']) return false;

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Clever",

},

ominouswind: {

num: 466,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 10% chance to raise the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage.",

shortDesc: "10% chance to raise all stats by 1 (not acc/eva).",

isNonstandard: "Past",

name: "Ominous Wind",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

self: {

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

},

},

target: "normal",

type: "Ghost",

contestType: "Beautiful",

},

originpulse: {

num: 618,

accuracy: 85,

basePower: 110,

category: "Special",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent foes.",

isNonstandard: "Past",

name: "Origin Pulse",

pp: 10,

priority: 0,

flags: {protect: 1, pulse: 1, mirror: 1},

target: "allAdjacentFoes",

type: "Water",

contestType: "Beautiful",

},

outrage: {

num: 200,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "The user spends two or three turns locked into this move and becomes confused immediately after its move on the last turn of the effect if it is not already. This move targets an opposing Pokemon at random on each turn. If the user is prevented from moving, is asleep at the beginning of a turn, or the attack is not successful against the target on the first turn of the effect or the second turn of a three-turn effect, the effect ends without causing confusion. If this move is called by Sleep Talk and the user is asleep, the move is used for one turn and does not confuse the user.",

shortDesc: "Lasts 2-3 turns. Confuses the user afterwards.",

name: "Outrage",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'lockedmove',

},

onAfterMove(pokemon) {

if (pokemon.volatiles['lockedmove'] && pokemon.volatiles['lockedmove'].duration === 1) {

pokemon.removeVolatile('lockedmove');

}

},

secondary: null,

target: "randomNormal",

type: "Dragon",

contestType: "Cool",

},

overdrive: {

num: 786,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits foe(s).",

name: "Overdrive",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Electric",

},

overheat: {

num: 315,

accuracy: 90,

basePower: 130,

category: "Special",

desc: "Lowers the user's Special Attack by 2 stages.",

shortDesc: "Lowers the user's Sp. Atk by 2.",

name: "Overheat",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

self: {

boosts: {

spa: -2,

},

},

secondary: null,

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

painsplit: {

num: 220,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and the target's HP become the average of their current HP, rounded down, but not more than the maximum HP of either one.",

shortDesc: "Shares HP of user and target equally.",

name: "Pain Split",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

onHit(target, pokemon) {

const targetHP = target.getUndynamaxedHP();

const averagehp = Math.floor((targetHP + pokemon.hp) / 2) || 1;

const targetChange = targetHP - averagehp;

target.sethp(target.hp - targetChange);

this.add('-sethp', target, target.getHealth, '[from] move: Pain Split', '[silent]');

pokemon.sethp(averagehp);

this.add('-sethp', pokemon, pokemon.getHealth, '[from] move: Pain Split');

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

paleowave: {

num: 0,

accuracy: 100,

basePower: 85,

category: "Special",

desc: "Has a 20% chance to lower the target's Attack by 1 stage.",

shortDesc: "20% chance to lower the target's Attack by 1.",

isNonstandard: "CAP",

name: "Paleo Wave",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

boosts: {

atk: -1,

},

},

target: "normal",

type: "Rock",

contestType: "Beautiful",

},

paraboliccharge: {

num: 570,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "The user recovers 1/2 the HP lost by the target, rounded half up. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down.",

shortDesc: "User recovers 50% of the damage dealt.",

name: "Parabolic Charge",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, heal: 1},

drain: [1, 2],

secondary: null,

target: "allAdjacent",

type: "Electric",

contestType: "Clever",

},

partingshot: {

num: 575,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack and Special Attack by 1 stage. If this move is successful, the user switches out even if it is trapped and is replaced immediately by a selected party member. The user does not switch out if the target's Attack and Special Attack stat stages were both unchanged, or if there are no unfainted party members.",

shortDesc: "Lowers target's Atk, Sp. Atk by 1. User switches.",

name: "Parting Shot",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

onHit(target, source, move) {

const success = this.boost({atk: -1, spa: -1}, target, source);

if (!success && !target.hasAbility('mirrorarmor')) {

delete move.selfSwitch;

}

},

selfSwitch: true,

secondary: null,

target: "normal",

type: "Dark",

zMove: {effect: 'healreplacement'},

contestType: "Cool",

},

payback: {

num: 371,

accuracy: 100,

basePower: 50,

basePowerCallback(pokemon, target, move) {

if (target.newlySwitched || this.queue.willMove(target)) {

this.debug('Payback NOT boosted');

return move.basePower;

}

this.debug('Payback damage boost');

return move.basePower \* 2;

},

category: "Physical",

desc: "Power doubles if the user moves after the target this turn, including actions taken through Instruct or the Dancer Ability. Switching in does not count as an action.",

shortDesc: "Power doubles if the user moves after the target.",

name: "Payback",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Tough",

},

payday: {

num: 6,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Scatters coins.",

name: "Pay Day",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit() {

this.add('-fieldactivate', 'move: Pay Day');

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Clever",

},

peck: {

num: 64,

accuracy: 100,

basePower: 35,

category: "Physical",

shortDesc: "No additional effect.",

name: "Peck",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

perishsong: {

num: 195,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Each active Pokemon receives a perish count of 4 if it doesn't already have a perish count. At the end of each turn including the turn used, the perish count of all active Pokemon lowers by 1 and Pokemon faint if the number reaches 0. The perish count is removed from Pokemon that switch out. If a Pokemon uses Baton Pass while it has a perish count, the replacement will gain the perish count and continue to count down.",

shortDesc: "All active Pokemon will faint in 3 turns.",

name: "Perish Song",

pp: 5,

priority: 0,

flags: {sound: 1, distance: 1, authentic: 1},

onHitField(target, source, move) {

let result = false;

let message = false;

for (const pokemon of this.getAllActive()) {

if (this.runEvent('Invulnerability', pokemon, source, move) === false) {

this.add('-miss', source, pokemon);

result = true;

} else if (this.runEvent('TryHit', pokemon, source, move) === null) {

result = true;

} else if (!pokemon.volatiles['perishsong']) {

pokemon.addVolatile('perishsong');

this.add('-start', pokemon, 'perish3', '[silent]');

result = true;

message = true;

}

}

if (!result) return false;

if (message) this.add('-fieldactivate', 'move: Perish Song');

},

condition: {

duration: 4,

onEnd(target) {

this.add('-start', target, 'perish0');

target.faint();

},

onResidualOrder: 20,

onResidual(pokemon) {

const duration = pokemon.volatiles['perishsong'].duration;

this.add('-start', pokemon, 'perish' + duration);

},

},

secondary: null,

target: "all",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

petalblizzard: {

num: 572,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent Pokemon.",

name: "Petal Blizzard",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "allAdjacent",

type: "Grass",

contestType: "Beautiful",

},

petaldance: {

num: 80,

accuracy: 100,

basePower: 120,

category: "Special",

desc: "The user spends two or three turns locked into this move and becomes confused immediately after its move on the last turn of the effect if it is not already. This move targets an opposing Pokemon at random on each turn. If the user is prevented from moving, is asleep at the beginning of a turn, or the attack is not successful against the target on the first turn of the effect or the second turn of a three-turn effect, the effect ends without causing confusion. If this move is called by Sleep Talk and the user is asleep, the move is used for one turn and does not confuse the user.",

shortDesc: "Lasts 2-3 turns. Confuses the user afterwards.",

name: "Petal Dance",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, dance: 1},

self: {

volatileStatus: 'lockedmove',

},

onAfterMove(pokemon) {

if (pokemon.volatiles['lockedmove'] && pokemon.volatiles['lockedmove'].duration === 1) {

pokemon.removeVolatile('lockedmove');

}

},

secondary: null,

target: "randomNormal",

type: "Grass",

contestType: "Beautiful",

},

phantomforce: {

num: 566,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "If this move is successful, it breaks through the target's Baneful Bunker, Detect, King's Shield, Protect, or Spiky Shield for this turn, allowing other Pokemon to attack the target normally. If the target's side is protected by Crafty Shield, Mat Block, Quick Guard, or Wide Guard, that protection is also broken for this turn and other Pokemon may attack the target's side normally. This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Disappears turn 1. Hits turn 2. Breaks protection.",

name: "Phantom Force",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, mirror: 1},

breaksProtect: true,

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onInvulnerability: false,

},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

photongeyser: {

num: 722,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "This move becomes a physical attack if the user's Attack is greater than its Special Attack, including stat stage changes. This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Physical if user's Atk > Sp. Atk. Ignores Abilities.",

name: "Photon Geyser",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move, pokemon) {

if (pokemon.getStat('atk', false, true) > pokemon.getStat('spa', false, true)) move.category = 'Physical';

},

ignoreAbility: true,

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

pikapapow: {

num: 732,

accuracy: true,

basePower: 0,

basePowerCallback(pokemon) {

return Math.floor((pokemon.happiness \* 10) / 25) || 1;

},

category: "Special",

desc: "Power is equal to the greater of (user's Happiness \* 2/5), rounded down, or 1.",

shortDesc: "Max happiness: 102 power. Can't miss.",

isNonstandard: "LGPE",

name: "Pika Papow",

pp: 20,

priority: 0,

flags: {protect: 1},

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cute",

},

pinmissile: {

num: 42,

accuracy: 95,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Pin Missile",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Bug",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Cool",

},

plasmafists: {

num: 721,

accuracy: 100,

basePower: 100,

category: "Physical",

desc: "If this move is successful, causes Normal-type moves to become Electric type this turn.",

shortDesc: "Normal moves become Electric type this turn.",

name: "Plasma Fists",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

pseudoWeather: 'iondeluge',

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

playnice: {

num: 589,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 1 stage.",

shortDesc: "Lowers the target's Attack by 1.",

name: "Play Nice",

pp: 20,

priority: 0,

flags: {reflectable: 1, mirror: 1, authentic: 1},

boosts: {

atk: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

playrough: {

num: 583,

accuracy: 90,

basePower: 90,

category: "Physical",

desc: "Has a 10% chance to lower the target's Attack by 1 stage.",

shortDesc: "10% chance to lower the target's Attack by 1.",

name: "Play Rough",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

atk: -1,

},

},

target: "normal",

type: "Fairy",

contestType: "Cute",

},

pluck: {

num: 365,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "If this move is successful and the user has not fainted, it steals the target's held Berry if it is holding one and eats it immediately, gaining its effects even if the user's item is being ignored. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "User steals and eats the target's Berry.",

name: "Pluck",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

onHit(target, source) {

const item = target.getItem();

if (source.hp && item.isBerry && target.takeItem(source)) {

this.add('-enditem', target, item.name, '[from] stealeat', '[move] Pluck', '[of] ' + source);

if (this.singleEvent('Eat', item, null, source, null, null)) {

this.runEvent('EatItem', source, null, null, item);

if (item.id === 'leppaberry') target.staleness = 'external';

}

if (item.onEat) source.ateBerry = true;

}

},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cute",

},

poisonfang: {

num: 305,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Has a 50% chance to badly poison the target.",

shortDesc: "50% chance to badly poison the target.",

name: "Poison Fang",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

status: 'tox',

},

target: "normal",

type: "Poison",

contestType: "Clever",

},

poisongas: {

num: 139,

accuracy: 90,

basePower: 0,

category: "Status",

desc: "Poisons the target.",

shortDesc: "Poisons the foe(s).",

name: "Poison Gas",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'psn',

secondary: null,

target: "allAdjacentFoes",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

poisonjab: {

num: 398,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 30% chance to poison the target.",

shortDesc: "30% chance to poison the target.",

name: "Poison Jab",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Tough",

},

poisonpowder: {

num: 77,

accuracy: 75,

basePower: 0,

category: "Status",

desc: "Poisons the target.",

shortDesc: "Poisons the target.",

name: "Poison Powder",

pp: 35,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1},

status: 'psn',

secondary: null,

target: "normal",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

poisonsting: {

num: 40,

accuracy: 100,

basePower: 15,

category: "Physical",

desc: "Has a 30% chance to poison the target.",

shortDesc: "30% chance to poison the target.",

name: "Poison Sting",

pp: 35,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Clever",

},

poisontail: {

num: 342,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "Has a 10% chance to poison the target and a higher chance for a critical hit.",

shortDesc: "High critical hit ratio. 10% chance to poison.",

name: "Poison Tail",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: {

chance: 10,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Clever",

},

pollenpuff: {

num: 676,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "If the target is an ally, this move restores 1/2 of its maximum HP, rounded down, instead of dealing damage.",

shortDesc: "If the target is an ally, heals 50% of its max HP.",

name: "Pollen Puff",

pp: 15,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

onTryHit(target, source, move) {

if (source.side === target.side) {

move.basePower = 0;

}

},

onHit(target, source) {

if (source.side === target.side) {

this.heal(Math.floor(target.baseMaxhp \* 0.5));

}

},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cute",

},

poltergeist: {

num: 809,

accuracy: 90,

basePower: 110,

category: "Physical",

shortDesc: "Fails if the target has no held item.",

name: "Poltergeist",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onTry(pokemon, target) {

if (!target.item) {

this.attrLastMove('[still]');

this.add('-fail', pokemon);

return null;

}

},

onTryHit(target, source, move) {

this.add('-activate', target, 'move: Poltergeist', this.dex.getItem(target.item).name);

},

secondary: null,

target: "normal",

type: "Ghost",

},

pound: {

num: 1,

accuracy: 100,

basePower: 40,

category: "Physical",

shortDesc: "No additional effect.",

name: "Pound",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

powder: {

num: 600,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "If the target uses a Fire-type move this turn, it is prevented from executing and the target loses 1/4 of its maximum HP, rounded half up. This effect does not happen if the Fire-type move is prevented by Primordial Sea.",

shortDesc: "If using a Fire move, target loses 1/4 max HP.",

isNonstandard: "Past",

name: "Powder",

pp: 20,

priority: 1,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'powder',

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'Powder');

},

onTryMovePriority: -1,

onTryMove(pokemon, target, move) {

if (move.type === 'Fire') {

this.add('-activate', pokemon, 'move: Powder');

this.damage(this.clampIntRange(Math.round(pokemon.maxhp / 4), 1));

return false;

}

},

},

secondary: null,

target: "normal",

type: "Bug",

zMove: {boost: {spd: 2}},

contestType: "Clever",

},

powdersnow: {

num: 181,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 10% chance to freeze the target.",

shortDesc: "10% chance to freeze the foe(s).",

name: "Powder Snow",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'frz',

},

target: "allAdjacentFoes",

type: "Ice",

contestType: "Beautiful",

},

powergem: {

num: 408,

accuracy: 100,

basePower: 80,

category: "Special",

shortDesc: "No additional effect.",

name: "Power Gem",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Beautiful",

},

powersplit: {

num: 471,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and the target have their Attack and Special Attack stats set to be equal to the average of the user and the target's Attack and Special Attack stats, respectively, rounded down. Stat stage changes are unaffected.",

shortDesc: "Averages Attack and Sp. Atk stats with target.",

name: "Power Split",

pp: 10,

priority: 0,

flags: {protect: 1, mystery: 1},

onHit(target, source) {

const newatk = Math.floor((target.storedStats.atk + source.storedStats.atk) / 2);

target.storedStats.atk = newatk;

source.storedStats.atk = newatk;

const newspa = Math.floor((target.storedStats.spa + source.storedStats.spa) / 2);

target.storedStats.spa = newspa;

source.storedStats.spa = newspa;

this.add('-activate', source, 'move: Power Split', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

powerswap: {

num: 384,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps its Attack and Special Attack stat stage changes with the target.",

shortDesc: "Swaps Attack and Sp. Atk stat stages with target.",

name: "Power Swap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, authentic: 1, mystery: 1},

onHit(target, source) {

const targetBoosts: SparseBoostsTable = {};

const sourceBoosts: SparseBoostsTable = {};

const atkSpa: BoostName[] = ['atk', 'spa'];

for (const stat of atkSpa) {

targetBoosts[stat] = target.boosts[stat];

sourceBoosts[stat] = source.boosts[stat];

}

source.setBoost(targetBoosts);

target.setBoost(sourceBoosts);

this.add('-swapboost', source, target, 'atk, spa', '[from] move: Power Swap');

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

powertrick: {

num: 379,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps its Attack and Defense stats, and stat stage changes remain on their respective stats. This move can be used again to swap the stats back. If the user uses Baton Pass, the replacement will have its Attack and Defense stats swapped if the effect is active. If the user has its stats recalculated by changing forme while its stats are swapped, this effect is ignored but is still active for the purposes of Baton Pass.",

shortDesc: "Switches user's Attack and Defense stats.",

name: "Power Trick",

pp: 10,

priority: 0,

flags: {snatch: 1},

volatileStatus: 'powertrick',

condition: {

onStart(pokemon) {

this.add('-start', pokemon, 'Power Trick');

const newatk = pokemon.storedStats.def;

const newdef = pokemon.storedStats.atk;

pokemon.storedStats.atk = newatk;

pokemon.storedStats.def = newdef;

},

onCopy(pokemon) {

const newatk = pokemon.storedStats.def;

const newdef = pokemon.storedStats.atk;

pokemon.storedStats.atk = newatk;

pokemon.storedStats.def = newdef;

},

onEnd(pokemon) {

this.add('-end', pokemon, 'Power Trick');

const newatk = pokemon.storedStats.def;

const newdef = pokemon.storedStats.atk;

pokemon.storedStats.atk = newatk;

pokemon.storedStats.def = newdef;

},

onRestart(pokemon) {

pokemon.removeVolatile('Power Trick');

},

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {boost: {atk: 1}},

contestType: "Clever",

},

powertrip: {

num: 681,

accuracy: 100,

basePower: 20,

basePowerCallback(pokemon, target, move) {

return move.basePower + 20 \* pokemon.positiveBoosts();

},

category: "Physical",

desc: "Power is equal to 20+(X\*20), where X is the user's total stat stage changes that are greater than 0.",

shortDesc: " + 20 power for each of the user's stat boosts.",

name: "Power Trip",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Clever",

},

poweruppunch: {

num: 612,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "Has a 100% chance to raise the user's Attack by 1 stage.",

shortDesc: "100% chance to raise the user's Attack by 1.",

name: "Power-Up Punch",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 100,

self: {

boosts: {

atk: 1,

},

},

},

target: "normal",

type: "Fighting",

contestType: "Tough",

},

powerwhip: {

num: 438,

accuracy: 85,

basePower: 120,

category: "Physical",

shortDesc: "No additional effect.",

name: "Power Whip",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Tough",

},

precipiceblades: {

num: 619,

accuracy: 85,

basePower: 120,

category: "Physical",

desc: "No additional effect.",

shortDesc: "No additional effect. Hits adjacent foes.",

isNonstandard: "Past",

name: "Precipice Blades",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

target: "allAdjacentFoes",

type: "Ground",

contestType: "Cool",

},

present: {

num: 217,

accuracy: 90,

basePower: 0,

category: "Physical",

desc: "If this move is successful, it deals damage or heals the target. 40% chance for 40 power, 30% chance for 80 power, 10% chance for 120 power, and 20% chance to heal the target by 1/4 of its maximum HP, rounded down.",

shortDesc: "40, 80, 120 power, or heals target 1/4 max HP.",

name: "Present",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move, pokemon, target) {

const rand = this.random(10);

if (rand < 2) {

move.heal = [1, 4];

} else if (rand < 6) {

move.basePower = 40;

} else if (rand < 9) {

move.basePower = 80;

} else {

move.basePower = 120;

}

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

prismaticlaser: {

num: 711,

accuracy: 100,

basePower: 160,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Prismatic Laser",

pp: 10,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

protect: {

num: 182,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Prevents moves from affecting the user this turn.",

name: "Protect",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'protect',

onPrepareHit(pokemon) {

return !!this.queue.willAct() && this.runEvent('StallMove', pokemon);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'Protect');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect']) {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Protect');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

return this.NOT\_FAIL;

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

psybeam: {

num: 60,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Has a 10% chance to confuse the target.",

shortDesc: "10% chance to confuse the target.",

name: "Psybeam",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'confusion',

},

target: "normal",

type: "Psychic",

contestType: "Beautiful",

},

psychup: {

num: 244,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user copies all of the target's current stat stage changes.",

shortDesc: "Copies the target's current stat stages.",

name: "Psych Up",

pp: 10,

priority: 0,

flags: {authentic: 1, mystery: 1},

onHit(target, source) {

let i: BoostName;

for (i in target.boosts) {

source.boosts[i] = target.boosts[i];

}

const volatilesToCopy = ['focusenergy', 'gmaxchistrike', 'laserfocus'];

for (const volatile of volatilesToCopy) {

if (target.volatiles[volatile]) {

source.addVolatile(volatile);

if (volatile === 'gmaxchistrike') source.volatiles[volatile].layers = target.volatiles[volatile].layers;

} else {

source.removeVolatile(volatile);

}

}

this.add('-copyboost', source, target, '[from] move: Psych Up');

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Clever",

},

psychic: {

num: 94,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "10% chance to lower the target's Sp. Def by 1.",

name: "Psychic",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Psychic",

contestType: "Clever",

},

psychicfangs: {

num: 706,

accuracy: 100,

basePower: 85,

category: "Physical",

desc: "If this attack does not miss, the effects of Reflect, Light Screen, and Aurora Veil end for the target's side of the field before damage is calculated.",

shortDesc: "Destroys screens, unless the target is immune.",

name: "Psychic Fangs",

pp: 10,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

onTryHit(pokemon) {

// will shatter screens through sub, before you hit

if (pokemon.runImmunity('Psychic')) {

pokemon.side.removeSideCondition('reflect');

pokemon.side.removeSideCondition('lightscreen');

pokemon.side.removeSideCondition('auroraveil');

}

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

psychicterrain: {

num: 678,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the terrain becomes Psychic Terrain. During the effect, the power of Psychic-type attacks made by grounded Pokemon is multiplied by 1.3 and grounded Pokemon cannot be hit by moves with priority greater than 0, unless the target is an ally. Camouflage transforms the user into a Psychic type, Nature Power becomes Psychic, and Secret Power has a 30% chance to lower the target's Speed by 1 stage. Fails if the current terrain is Psychic Terrain.",

shortDesc: "5 turns. Grounded: +Psychic power, priority-safe.",

name: "Psychic Terrain",

pp: 10,

priority: 0,

flags: {nonsky: 1},

terrain: 'psychicterrain',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasItem('terrainextender')) {

return 8;

}

return 5;

},

onTryHitPriority: 4,

onTryHit(target, source, effect) {

if (effect && (effect.priority <= 0.1 || effect.target === 'self')) {

return;

}

if (target.isSemiInvulnerable() || target.side === source.side) return;

if (!target.isGrounded()) {

const baseMove = this.dex.getMove(effect.id);

if (baseMove.priority > 0) {

this.hint("Psychic Terrain doesn't affect Pokémon immune to Ground.");

}

return;

}

this.add('-activate', target, 'move: Psychic Terrain');

return null;

},

onBasePowerPriority: 6,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Psychic' && attacker.isGrounded() && !attacker.isSemiInvulnerable()) {

this.debug('psychic terrain boost');

return this.chainModify([0x14CD, 0x1000]);

}

},

onStart(battle, source, effect) {

if (effect?.effectType === 'Ability') {

this.add('-fieldstart', 'move: Psychic Terrain', '[from] ability: ' + effect, '[of] ' + source);

} else {

this.add('-fieldstart', 'move: Psychic Terrain');

}

},

onResidualOrder: 21,

onResidualSubOrder: 2,

onEnd() {

this.add('-fieldend', 'move: Psychic Terrain');

},

},

secondary: null,

target: "all",

type: "Psychic",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

psychoboost: {

num: 354,

accuracy: 90,

basePower: 140,

category: "Special",

desc: "Lowers the user's Special Attack by 2 stages.",

shortDesc: "Lowers the user's Sp. Atk by 2.",

isNonstandard: "Past",

name: "Psycho Boost",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

self: {

boosts: {

spa: -2,

},

},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

psychocut: {

num: 427,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Psycho Cut",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

psychoshift: {

num: 375,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "The user's major status condition is transferred to the target, and the user is then cured. Fails if the user has no major status condition or if the target already has one.",

shortDesc: "Transfers the user's status ailment to the target.",

name: "Psycho Shift",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onPrepareHit(target, source, move) {

if (!source.status) return false;

move.status = source.status;

},

self: {

onHit(pokemon) {

pokemon.cureStatus();

},

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spa: 2}},

contestType: "Clever",

},

psyshock: {

num: 473,

accuracy: 100,

basePower: 80,

category: "Special",

defensiveCategory: "Physical",

desc: "Deals damage to the target based on its Defense instead of Special Defense.",

shortDesc: "Damages target based on Defense, not Sp. Def.",

name: "Psyshock",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Beautiful",

},

psystrike: {

num: 540,

accuracy: 100,

basePower: 100,

category: "Special",

defensiveCategory: "Physical",

desc: "Deals damage to the target based on its Defense instead of Special Defense.",

shortDesc: "Damages target based on Defense, not Sp. Def.",

name: "Psystrike",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

psywave: {

num: 149,

accuracy: 100,

basePower: 0,

damageCallback(pokemon) {

return (this.random(50, 151) \* pokemon.level) / 100;

},

category: "Special",

desc: "Deals damage to the target equal to (user's level) \* (X + 50) / 100, where X is a random number from 0 to 100, rounded down, but not less than 1 HP.",

shortDesc: "Random damage equal to 0.5x-1.5x user's level.",

isNonstandard: "Past",

name: "Psywave",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Clever",

},

pulverizingpancake: {

num: 701,

accuracy: true,

basePower: 210,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Pulverizing Pancake",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "snorliumz",

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

punishment: {

num: 386,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

let power = 60 + 20 \* target.positiveBoosts();

if (power > 200) power = 200;

return power;

},

category: "Physical",

desc: "Power is equal to 60+(X\*20), where X is the target's total stat stage changes that are greater than 0, but not more than 200 power.",

shortDesc: "60 power +20 for each of the target's stat boosts.",

isNonstandard: "Past",

name: "Punishment",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Dark",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cool",

},

purify: {

num: 685,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target is cured if it has a major status condition. If the target was cured, the user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Cures target's status; heals user 1/2 max HP if so.",

name: "Purify",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, heal: 1},

onHit(target, source) {

if (!target.cureStatus()) return false;

this.heal(Math.ceil(source.maxhp \* 0.5), source);

},

secondary: null,

target: "normal",

type: "Poison",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Beautiful",

},

pursuit: {

num: 228,

accuracy: 100,

basePower: 40,

basePowerCallback(pokemon, target, move) {

// You can't get here unless the pursuit succeeds

if (target.beingCalledBack) {

this.debug('Pursuit damage boost');

return move.basePower \* 2;

}

return move.basePower;

},

category: "Physical",

desc: "If an opposing Pokemon switches out this turn, this move hits that Pokemon before it leaves the field, even if it was not the original target. If the user moves after an opponent using Parting Shot, U-turn, or Volt Switch, but not Baton Pass, it will hit that opponent before it leaves the field. Power doubles and no accuracy check is done if the user hits an opponent switching out, and the user's turn is over; if an opponent faints from this, the replacement Pokemon does not become active until the end of the turn.",

shortDesc: "If a foe is switching out, hits it at 2x power.",

isNonstandard: "Past",

name: "Pursuit",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

beforeTurnCallback(pokemon) {

for (const side of this.sides) {

if (side === pokemon.side) continue;

side.addSideCondition('pursuit', pokemon);

const data = side.getSideConditionData('pursuit');

if (!data.sources) {

data.sources = [];

}

data.sources.push(pokemon);

}

},

onModifyMove(move, source, target) {

if (target?.beingCalledBack) move.accuracy = true;

},

onTryHit(target, pokemon) {

target.side.removeSideCondition('pursuit');

},

condition: {

duration: 1,

onBeforeSwitchOut(pokemon) {

this.debug('Pursuit start');

let alreadyAdded = false;

pokemon.removeVolatile('destinybond');

for (const source of this.effectData.sources) {

if (!this.queue.cancelMove(source) || !source.hp) continue;

if (!alreadyAdded) {

this.add('-activate', pokemon, 'move: Pursuit');

alreadyAdded = true;

}

// Run through each action in queue to check if the Pursuit user is supposed to Mega Evolve this turn.

// If it is, then Mega Evolve before moving.

if (source.canMegaEvo || source.canUltraBurst) {

for (const [actionIndex, action] of this.queue.entries()) {

if (action.pokemon === source && action.choice === 'megaEvo') {

this.runMegaEvo(source);

this.queue.list.splice(actionIndex, 1);

break;

}

}

}

this.runMove('pursuit', source, this.getTargetLoc(pokemon, source));

}

},

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

pyroball: {

num: 780,

accuracy: 90,

basePower: 120,

category: "Physical",

desc: "Has a 10% chance to burn the target.",

shortDesc: "10% chance to burn the target. Thaws user.",

name: "Pyro Ball",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1, bullet: 1},

secondary: {

chance: 10,

status: 'brn',

},

target: "normal",

type: "Fire",

},

quash: {

num: 511,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to take its turn after all other Pokemon this turn, no matter the priority of its selected move. Fails if the target already moved this turn.",

shortDesc: "Forces the target to move last this turn.",

name: "Quash",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

onHit(target) {

if (target.side.active.length < 2) return false; // fails in singles

const action = this.queue.willMove(target);

if (!action) return false;

action.order = 201;

this.add('-activate', target, 'move: Quash');

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

quickattack: {

num: 98,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Quick Attack",

pp: 30,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

quickguard: {

num: 501,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and its party members are protected from attacks with original or altered priority greater than 0 made by other Pokemon, including allies, during this turn. This move modifies the same 1/X chance of being successful used by other protection moves, where X starts at 1 and triples each time this move is successfully used, but does not use the chance to check for failure. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn or if this move is already in effect for the user's side.",

shortDesc: "Protects allies from priority attacks this turn.",

name: "Quick Guard",

pp: 15,

priority: 3,

flags: {snatch: 1},

sideCondition: 'quickguard',

onTryHitSide(side, source) {

return !!this.queue.willAct();

},

onHitSide(side, source) {

source.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target, source) {

this.add('-singleturn', source, 'Quick Guard');

},

onTryHitPriority: 4,

onTryHit(target, source, move) {

// Quick Guard blocks moves with positive priority, even those given increased priority by Prankster or Gale Wings.

// (e.g. it blocks 0 priority moves boosted by Prankster or Gale Wings; Quick Claw/Custap Berry do not count)

if (move.priority <= 0.1) return;

if (!move.flags['protect']) {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

this.add('-activate', target, 'move: Quick Guard');

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

return this.NOT\_FAIL;

},

},

secondary: null,

target: "allySide",

type: "Fighting",

zMove: {boost: {def: 1}},

contestType: "Cool",

},

quiverdance: {

num: 483,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Attack, Special Defense, and Speed by 1 stage.",

shortDesc: "Raises the user's Sp. Atk, Sp. Def, Speed by 1.",

name: "Quiver Dance",

pp: 20,

priority: 0,

flags: {snatch: 1, dance: 1},

boosts: {

spa: 1,

spd: 1,

spe: 1,

},

secondary: null,

target: "self",

type: "Bug",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

rage: {

num: 99,

accuracy: 100,

basePower: 20,

category: "Physical",

desc: "Once this move is successfully used, the user's Attack is raised by 1 stage every time it is hit by another Pokemon's attack as long as this move is chosen for use.",

shortDesc: "Raises the user's Attack by 1 if hit during use.",

isNonstandard: "Past",

name: "Rage",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'rage',

},

condition: {

onStart(pokemon) {

this.add('-singlemove', pokemon, 'Rage');

},

onHit(target, source, move) {

if (target !== source && move.category !== 'Status') {

this.boost({atk: 1});

}

},

onBeforeMovePriority: 100,

onBeforeMove(pokemon) {

this.debug('removing Rage before attack');

pokemon.removeVolatile('rage');

},

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

ragepowder: {

num: 476,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the turn, all single-target attacks from the opposing side are redirected to the user. Such attacks are redirected to the user before they can be reflected by Magic Coat or the Magic Bounce Ability, or drawn in by the Lightning Rod or Storm Drain Abilities. Fails if it is not a Double Battle or Battle Royal. This effect is ignored while the user is under the effect of Sky Drop.",

shortDesc: "The foes' moves target the user on the turn used.",

name: "Rage Powder",

pp: 20,

priority: 2,

flags: {powder: 1},

volatileStatus: 'ragepowder',

onTryHit(target) {

if (target.side.active.length < 2) return false;

},

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'move: Rage Powder');

},

onFoeRedirectTargetPriority: 1,

onFoeRedirectTarget(target, source, source2, move) {

const ragePowderUser = this.effectData.target;

if (ragePowderUser.isSkyDropped()) return;

if (source.runStatusImmunity('powder') && this.validTarget(ragePowderUser, source, move.target)) {

if (move.smartTarget) move.smartTarget = false;

this.debug("Rage Powder redirected target of move");

return ragePowderUser;

}

},

},

secondary: null,

target: "self",

type: "Bug",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

raindance: {

num: 240,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the weather becomes Rain Dance. The damage of Water-type attacks is multiplied by 1.5 and the damage of Fire-type attacks is multiplied by 0.5 during the effect. Lasts for 8 turns if the user is holding Damp Rock. Fails if the current weather is Rain Dance.",

shortDesc: "For 5 turns, heavy rain powers Water moves.",

name: "Rain Dance",

pp: 5,

priority: 0,

flags: {},

weather: 'RainDance',

secondary: null,

target: "all",

type: "Water",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

rapidspin: {

num: 229,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "If this move is successful and the user has not fainted, the effects of Leech Seed and binding moves end for the user, and all hazards are removed from the user's side of the field. Has a 100% chance to raise the user's Speed by 1 stage.",

shortDesc: "Free user from hazards/bind/Leech Seed; +1 Spe.",

name: "Rapid Spin",

pp: 40,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onAfterHit(target, pokemon) {

if (pokemon.hp && pokemon.removeVolatile('leechseed')) {

this.add('-end', pokemon, 'Leech Seed', '[from] move: Rapid Spin', '[of] ' + pokemon);

}

const sideConditions = ['spikes', 'toxicspikes', 'stealthrock', 'stickyweb', 'gmaxsteelsurge'];

for (const condition of sideConditions) {

if (pokemon.hp && pokemon.side.removeSideCondition(condition)) {

this.add('-sideend', pokemon.side, this.dex.getEffect(condition).name, '[from] move: Rapid Spin', '[of] ' + pokemon);

}

}

if (pokemon.hp && pokemon.volatiles['partiallytrapped']) {

pokemon.removeVolatile('partiallytrapped');

}

},

onAfterSubDamage(damage, target, pokemon) {

if (pokemon.hp && pokemon.removeVolatile('leechseed')) {

this.add('-end', pokemon, 'Leech Seed', '[from] move: Rapid Spin', '[of] ' + pokemon);

}

const sideConditions = ['spikes', 'toxicspikes', 'stealthrock', 'stickyweb', 'gmaxsteelsurge'];

for (const condition of sideConditions) {

if (pokemon.hp && pokemon.side.removeSideCondition(condition)) {

this.add('-sideend', pokemon.side, this.dex.getEffect(condition).name, '[from] move: Rapid Spin', '[of] ' + pokemon);

}

}

if (pokemon.hp && pokemon.volatiles['partiallytrapped']) {

pokemon.removeVolatile('partiallytrapped');

}

},

secondary: {

chance: 100,

self: {

boosts: {

spe: 1,

},

},

},

target: "normal",

type: "Normal",

contestType: "Cool",

},

razorleaf: {

num: 75,

accuracy: 95,

basePower: 55,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio. Hits adjacent foes.",

name: "Razor Leaf",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "allAdjacentFoes",

type: "Grass",

contestType: "Cool",

},

razorshell: {

num: 534,

accuracy: 95,

basePower: 75,

category: "Physical",

desc: "Has a 50% chance to lower the target's Defense by 1 stage.",

shortDesc: "50% chance to lower the target's Defense by 1.",

name: "Razor Shell",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

def: -1,

},

},

target: "normal",

type: "Water",

contestType: "Cool",

},

razorwind: {

num: 13,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a higher chance for a critical hit. This attack charges on the first turn and executes on the second. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Charges, then hits foe(s) turn 2. High crit ratio.",

isNonstandard: "Past",

name: "Razor Wind",

pp: 10,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

critRatio: 2,

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

contestType: "Cool",

},

recover: {

num: 105,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Heals the user by 50% of its max HP.",

name: "Recover",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

recycle: {

num: 278,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user regains the item it last used. Fails if the user is holding an item, if the user has not held an item, if the item was a popped Air Balloon, if the item was picked up by a Pokemon with the Pickup Ability, or if the item was lost to Bug Bite, Covet, Incinerate, Knock Off, Pluck, or Thief. Items thrown with Fling can be regained.",

shortDesc: "Restores the item the user last used.",

name: "Recycle",

pp: 10,

priority: 0,

flags: {snatch: 1},

onHit(pokemon) {

if (pokemon.item || !pokemon.lastItem) return false;

const item = pokemon.lastItem;

pokemon.lastItem = '';

this.add('-item', pokemon, this.dex.getItem(item), '[from] move: Recycle');

pokemon.setItem(item);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

reflect: {

num: 115,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members take 0.5x damage from physical attacks, or 0.66x damage if in a Double Battle. Damage is not reduced further with Aurora Veil. Critical hits ignore this effect. It is removed from the user's side if the user or an ally is successfully hit by Brick Break, Psychic Fangs, or Defog. Lasts for 8 turns if the user is holding Light Clay. Fails if the effect is already active on the user's side.",

shortDesc: "For 5 turns, physical damage to allies is halved.",

name: "Reflect",

pp: 20,

priority: 0,

flags: {snatch: 1},

sideCondition: 'reflect',

condition: {

duration: 5,

durationCallback(target, source, effect) {

if (source?.hasItem('lightclay')) {

return 8;

}

return 5;

},

onAnyModifyDamage(damage, source, target, move) {

if (target !== source && target.side === this.effectData.target && this.getCategory(move) === 'Physical') {

if (!target.getMoveHitData(move).crit && !move.infiltrates) {

this.debug('Reflect weaken');

if (target.side.active.length > 1) return this.chainModify([0xAAC, 0x1000]);

return this.chainModify(0.5);

}

}

},

onStart(side) {

this.add('-sidestart', side, 'Reflect');

},

onResidualOrder: 21,

onEnd(side) {

this.add('-sideend', side, 'Reflect');

},

},

secondary: null,

target: "allySide",

type: "Psychic",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

reflecttype: {

num: 513,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Causes the user's types to become the same as the current types of the target. If the target's current types include typeless and a non-added type, typeless is ignored. If the target's current types include typeless and an added type from Forest's Curse or Trick-or-Treat, typeless is copied as the Normal type instead. Fails if the user is an Arceus or a Silvally, or if the target's current type is typeless alone.",

shortDesc: "User becomes the same type as the target.",

name: "Reflect Type",

pp: 15,

priority: 0,

flags: {protect: 1, authentic: 1, mystery: 1},

onHit(target, source) {

if (source.species && (source.species.num === 493 || source.species.num === 773)) return false;

let newBaseTypes = target.getTypes(true).filter(type => type !== '???');

if (!newBaseTypes.length) {

if (target.addedType) {

newBaseTypes = ['Normal'];

} else {

return false;

}

}

this.add('-start', source, 'typechange', '[from] move: Reflect Type', '[of] ' + target);

source.setType(newBaseTypes);

source.addedType = target.addedType;

source.knownType = target.side === source.side && target.knownType;

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

refresh: {

num: 287,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user cures its burn, poison, or paralysis. Fails if the user is not burned, poisoned, or paralyzed.",

shortDesc: "User cures its burn, poison, or paralysis.",

isNonstandard: "Past",

name: "Refresh",

pp: 20,

priority: 0,

flags: {snatch: 1},

onHit(pokemon) {

if (['', 'slp', 'frz'].includes(pokemon.status)) return false;

pokemon.cureStatus();

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Cute",

},

relicsong: {

num: 547,

accuracy: 100,

basePower: 75,

category: "Special",

desc: "Has a 10% chance to cause the target to fall asleep. If this move is successful on at least one target and the user is a Meloetta, it changes to Pirouette Forme if it is currently in Aria Forme, or changes to Aria Forme if it is currently in Pirouette Forme. This forme change does not happen if the Meloetta has the Sheer Force Ability. The Pirouette Forme reverts to Aria Forme when Meloetta is not active.",

shortDesc: "10% chance to sleep foe(s). Meloetta transforms.",

isNonstandard: "Past",

name: "Relic Song",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: {

chance: 10,

status: 'slp',

},

onHit(target, pokemon, move) {

if (pokemon.baseSpecies.baseSpecies === 'Meloetta' && !pokemon.transformed) {

move.willChangeForme = true;

}

},

onAfterMoveSecondarySelf(pokemon, target, move) {

if (move.willChangeForme) {

const meloettaForme = pokemon.species.id === 'meloettapirouette' ? '' : '-Pirouette';

pokemon.formeChange('Meloetta' + meloettaForme, this.effect, false, '[msg]');

}

},

target: "allAdjacentFoes",

type: "Normal",

contestType: "Beautiful",

},

rest: {

num: 156,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user falls asleep for the next two turns and restores all of its HP, curing itself of any major status condition in the process. Fails if the user has full HP, is already asleep, or if another effect is preventing sleep.",

shortDesc: "User sleeps 2 turns and restores HP and status.",

name: "Rest",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

onTryMove(pokemon) {

if (pokemon.hp === pokemon.maxhp) {

this.add('-fail', pokemon, 'heal');

return null;

}

if (pokemon.status === 'slp' || pokemon.hasAbility('comatose')) {

this.add('-fail', pokemon);

return null;

}

},

onHit(target, source, move) {

if (!target.setStatus('slp', source, move)) return false;

target.statusData.time = 3;

target.statusData.startTime = 3;

this.heal(target.maxhp); // Aesthetic only as the healing happens after you fall asleep in-game

},

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

retaliate: {

num: 514,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Power doubles if one of the user's party members fainted last turn.",

shortDesc: "Power doubles if an ally fainted last turn.",

name: "Retaliate",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onBasePower(basePower, pokemon) {

if (pokemon.side.faintedLastTurn) {

this.debug('Boosted for a faint last turn');

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

return: {

num: 216,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon) {

return Math.floor((pokemon.happiness \* 10) / 25) || 1;

},

category: "Physical",

desc: "Power is equal to the greater of (user's Happiness \* 2/5), rounded down, or 1.",

shortDesc: "Max 102 power at maximum Happiness.",

isNonstandard: "Past",

name: "Return",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cute",

},

revelationdance: {

num: 686,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "This move's type depends on the user's primary type. If the user's primary type is typeless, this move's type is the user's secondary type if it has one, otherwise the added type from Forest's Curse or Trick-or-Treat. This move is typeless if the user's type is typeless alone.",

shortDesc: "Type varies based on the user's primary type.",

isNonstandard: "Past",

name: "Revelation Dance",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, dance: 1},

onModifyType(move, pokemon) {

let type = pokemon.types[0];

if (type === "Bird") type = "???";

move.type = type;

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

revenge: {

num: 279,

accuracy: 100,

basePower: 60,

basePowerCallback(pokemon, target, move) {

const damagedByTarget = pokemon.attackedBy.some(

p => p.source === target && p.damage > 0 && p.thisTurn

);

if (damagedByTarget) {

this.debug('Boosted for getting hit by ' + target);

return move.basePower \* 2;

}

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the user was hit by the target this turn.",

shortDesc: "Power doubles if user is damaged by the target.",

name: "Revenge",

pp: 10,

priority: -4,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

reversal: {

num: 179,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

const ratio = pokemon.hp \* 48 / pokemon.maxhp;

if (ratio < 2) {

return 200;

}

if (ratio < 5) {

return 150;

}

if (ratio < 10) {

return 100;

}

if (ratio < 17) {

return 80;

}

if (ratio < 33) {

return 40;

}

return 20;

},

category: "Physical",

desc: "The power of this move is 20 if X is 33 to 48, 40 if X is 17 to 32, 80 if X is 10 to 16, 100 if X is 5 to 9, 150 if X is 2 to 4, and 200 if X is 0 or 1, where X is equal to (user's current HP \* 48 / user's maximum HP), rounded down.",

shortDesc: "More power the less HP the user has left.",

name: "Reversal",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

zMove: {basePower: 160},

contestType: "Cool",

},

risingvoltage: {

num: 804,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "If the current terrain is Electric Terrain and the target is grounded, this move's power is doubled.",

shortDesc: "2x power if target is grounded in Electric Terrain.",

name: "Rising Voltage",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower, pokemon, target) {

if (this.field.isTerrain('electricterrain') && target.isGrounded()) {

this.debug('terrain buff');

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Electric",

},

roar: {

num: 46,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target is forced to switch out and be replaced with a random unfainted ally. Fails if the target is the last unfainted Pokemon in its party, or if the target used Ingrain previously or has the Suction Cups Ability.",

shortDesc: "Forces the target to switch to a random ally.",

name: "Roar",

pp: 20,

priority: -6,

flags: {reflectable: 1, mirror: 1, sound: 1, authentic: 1, mystery: 1},

forceSwitch: true,

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cool",

},

roaroftime: {

num: 459,

accuracy: 90,

basePower: 150,

category: "Special",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

isNonstandard: "Past",

name: "Roar of Time",

pp: 5,

priority: 0,

flags: {recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Beautiful",

},

rockblast: {

num: 350,

accuracy: 90,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Rock Blast",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Rock",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Tough",

},

rockclimb: {

num: 431,

accuracy: 85,

basePower: 90,

category: "Physical",

desc: "Has a 20% chance to confuse the target.",

shortDesc: "20% chance to confuse the target.",

isNonstandard: "Past",

name: "Rock Climb",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'confusion',

},

target: "normal",

type: "Normal",

contestType: "Tough",

},

rockpolish: {

num: 397,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Speed by 2 stages.",

shortDesc: "Raises the user's Speed by 2.",

name: "Rock Polish",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

spe: 2,

},

secondary: null,

target: "self",

type: "Rock",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

rockslide: {

num: 157,

accuracy: 90,

basePower: 75,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the foe(s).",

name: "Rock Slide",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "allAdjacentFoes",

type: "Rock",

contestType: "Tough",

},

rocksmash: {

num: 249,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "Has a 50% chance to lower the target's Defense by 1 stage.",

shortDesc: "50% chance to lower the target's Defense by 1.",

name: "Rock Smash",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

def: -1,

},

},

target: "normal",

type: "Fighting",

contestType: "Tough",

},

rockthrow: {

num: 88,

accuracy: 90,

basePower: 50,

category: "Physical",

shortDesc: "No additional effect.",

name: "Rock Throw",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Tough",

},

rocktomb: {

num: 317,

accuracy: 95,

basePower: 60,

category: "Physical",

desc: "Has a 100% chance to lower the target's Speed by 1 stage.",

shortDesc: "100% chance to lower the target's Speed by 1.",

name: "Rock Tomb",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spe: -1,

},

},

target: "normal",

type: "Rock",

contestType: "Clever",

},

rockwrecker: {

num: 439,

accuracy: 90,

basePower: 150,

category: "Physical",

desc: "If this move is successful, the user must recharge on the following turn and cannot select a move.",

shortDesc: "User cannot move next turn.",

name: "Rock Wrecker",

pp: 5,

priority: 0,

flags: {bullet: 1, recharge: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'mustrecharge',

},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Tough",

},

roleplay: {

num: 272,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user's Ability changes to match the target's Ability. Fails if the user's Ability is Battle Bond, Comatose, Disguise, Multitype, Power Construct, RKS System, Schooling, Shields Down, Stance Change, or already matches the target, or if the target's Ability is Battle Bond, Comatose, Disguise, Flower Gift, Forecast, Illusion, Imposter, Multitype, Neutralizing Gas, Power Construct, Power of Alchemy, Receiver, RKS System, Schooling, Shields Down, Stance Change, Trace, Wonder Guard, or Zen Mode.",

shortDesc: "User replaces its Ability with the target's.",

name: "Role Play",

pp: 10,

priority: 0,

flags: {authentic: 1, mystery: 1},

onTryHit(target, source) {

if (target.ability === source.ability) return false;

const bannedTargetAbilities = [

'battlebond', 'comatose', 'disguise', 'flowergift', 'forecast', 'illusion', 'imposter', 'multitype', 'neutralizinggas', 'powerconstruct', 'powerofalchemy', 'receiver', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'trace', 'wonderguard', 'zenmode',

];

const bannedSourceAbilities = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange',

];

if (bannedTargetAbilities.includes(target.ability) || bannedSourceAbilities.includes(source.ability)) {

return false;

}

},

onHit(target, source) {

const oldAbility = source.setAbility(target.ability);

if (oldAbility) {

this.add('-ability', source, source.getAbility().name, '[from] move: Role Play', '[of] ' + target);

return;

}

return false;

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Cute",

},

rollingkick: {

num: 27,

accuracy: 85,

basePower: 60,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

isNonstandard: "Past",

name: "Rolling Kick",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Fighting",

contestType: "Cool",

},

rollout: {

num: 205,

accuracy: 90,

basePower: 30,

basePowerCallback(pokemon, target, move) {

let bp = move.basePower;

if (pokemon.volatiles.rollout && pokemon.volatiles.rollout.hitCount) {

bp \*= Math.pow(2, pokemon.volatiles.rollout.hitCount);

}

if (pokemon.status !== 'slp') pokemon.addVolatile('rollout');

if (pokemon.volatiles.defensecurl) {

bp \*= 2;

}

this.debug("Rollout bp: " + bp);

return bp;

},

category: "Physical",

desc: "If this move is successful, the user is locked into this move and cannot make another move until it misses, 5 turns have passed, or the attack cannot be used. Power doubles with each successful hit of this move and doubles again if Defense Curl was used previously by the user. If this move is called by Sleep Talk, the move is used for one turn. If this move hits an active Disguise during the effect, the power multiplier is paused but the turn counter is not, potentially allowing the multiplier to be used on the user's next move after this effect ends.",

shortDesc: "Power doubles with each hit. Repeats for 5 turns.",

name: "Rollout",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

condition: {

duration: 2,

onLockMove: 'rollout',

onStart() {

this.effectData.hitCount = 1;

},

onRestart() {

this.effectData.hitCount++;

if (this.effectData.hitCount < 5) {

this.effectData.duration = 2;

}

},

onResidual(target) {

if (target.lastMove && target.lastMove.id === 'struggle') {

// don't lock

delete target.volatiles['rollout'];

}

},

},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Cute",

},

roost: {

num: 355,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up. Until the end of the turn, Flying-type users lose their Flying type and pure Flying-type users become Normal type. Does nothing if the user's HP is full.",

shortDesc: "Heals 50% HP. Flying-type removed 'til turn ends.",

name: "Roost",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

self: {

volatileStatus: 'roost',

},

condition: {

duration: 1,

onResidualOrder: 20,

onStart(target) {

this.add('-singleturn', target, 'move: Roost');

},

onTypePriority: -1,

onType(types, pokemon) {

this.effectData.typeWas = types;

return types.filter(type => type !== 'Flying');

},

},

secondary: null,

target: "self",

type: "Flying",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

rototiller: {

num: 563,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the Attack and Special Attack of all grounded Grass-type Pokemon on the field by 1 stage.",

shortDesc: "Raises Atk/Sp. Atk of grounded Grass types by 1.",

isNonstandard: "Past",

name: "Rototiller",

pp: 10,

priority: 0,

flags: {distance: 1, nonsky: 1},

onHitField(target, source) {

const targets: Pokemon[] = [];

let anyAirborne = false;

for (const pokemon of this.getAllActive()) {

if (!pokemon.runImmunity('Ground')) {

this.add('-immune', pokemon);

anyAirborne = true;

continue;

}

if (pokemon.hasType('Grass')) {

// This move affects every grounded Grass-type Pokemon in play.

targets.push(pokemon);

}

}

if (!targets.length && !anyAirborne) return false; // Fails when there are no grounded Grass types or airborne Pokemon

for (const pokemon of targets) {

this.boost({atk: 1, spa: 1}, pokemon, source);

}

},

secondary: null,

target: "all",

type: "Ground",

zMove: {boost: {atk: 1}},

contestType: "Tough",

},

round: {

num: 496,

accuracy: 100,

basePower: 60,

basePowerCallback(target, source, move) {

if (move.sourceEffect === 'round') {

return move.basePower \* 2;

}

return move.basePower;

},

category: "Special",

desc: "If there are other active Pokemon that chose this move for use this turn, those Pokemon take their turn immediately after the user, in Speed order, and this move's power is 120 for each other user.",

shortDesc: "Power doubles if others used Round this turn.",

name: "Round",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

onTry(target, source, move) {

for (const action of this.queue) {

// @ts-ignore

if (!action.pokemon || !action.move || action.maxMove || action.zmove) continue;

// @ts-ignore

if (action.move?.id === 'round') {

// @ts-ignore

this.queue.prioritizeAction(action, move);

return;

}

}

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

sacredfire: {

num: 221,

accuracy: 95,

basePower: 100,

category: "Physical",

desc: "Has a 50% chance to burn the target.",

shortDesc: "50% chance to burn the target. Thaws user.",

isNonstandard: "Past",

name: "Sacred Fire",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

secondary: {

chance: 50,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Beautiful",

},

sacredsword: {

num: 533,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "Ignores the target's stat stage changes, including evasiveness.",

shortDesc: "Ignores the target's stat stage changes.",

name: "Sacred Sword",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ignoreEvasion: true,

ignoreDefensive: true,

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

safeguard: {

num: 219,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the user and its party members cannot have major status conditions or confusion inflicted on them by other Pokemon. It is removed from the user's side if the user or an ally is successfully hit by Defog. Fails if the effect is already active on the user's side.",

shortDesc: "For 5 turns, protects user's party from status.",

name: "Safeguard",

pp: 25,

priority: 0,

flags: {snatch: 1},

sideCondition: 'safeguard',

condition: {

duration: 5,

durationCallback(target, source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onSetStatus(status, target, source, effect) {

if (!effect || !source) return;

if (effect.id === 'yawn') return;

if (effect.effectType === 'Move' && effect.infiltrates && target.side !== source.side) return;

if (target !== source) {

this.debug('interrupting setStatus');

if (effect.id === 'synchronize' || (effect.effectType === 'Move' && !effect.secondaries)) {

this.add('-activate', target, 'move: Safeguard');

}

return null;

}

},

onTryAddVolatile(status, target, source, effect) {

if (!effect || !source) return;

if (effect.effectType === 'Move' && effect.infiltrates && target.side !== source.side) return;

if ((status.id === 'confusion' || status.id === 'yawn') && target !== source) {

if (effect.effectType === 'Move' && !effect.secondaries) this.add('-activate', target, 'move: Safeguard');

return null;

}

},

onStart(side) {

this.add('-sidestart', side, 'Safeguard');

},

onResidualOrder: 21,

onResidualSubOrder: 2,

onEnd(side) {

this.add('-sideend', side, 'Safeguard');

},

},

secondary: null,

target: "allySide",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

sandattack: {

num: 28,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's accuracy by 1 stage.",

shortDesc: "Lowers the target's accuracy by 1.",

name: "Sand Attack",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

accuracy: -1,

},

secondary: null,

target: "normal",

type: "Ground",

zMove: {boost: {evasion: 1}},

contestType: "Cute",

},

sandstorm: {

num: 201,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the weather becomes Sandstorm. At the end of each turn except the last, all active Pokemon lose 1/16 of their maximum HP, rounded down, unless they are a Ground, Rock, or Steel type, or have the Magic Guard, Overcoat, Sand Force, Sand Rush, or Sand Veil Abilities. During the effect, the Special Defense of Rock-type Pokemon is multiplied by 1.5 when taking damage from a special attack. Lasts for 8 turns if the user is holding Smooth Rock. Fails if the current weather is Sandstorm.",

shortDesc: "For 5 turns, a sandstorm rages.",

name: "Sandstorm",

pp: 10,

priority: 0,

flags: {},

weather: 'Sandstorm',

secondary: null,

target: "all",

type: "Rock",

zMove: {boost: {spe: 1}},

contestType: "Tough",

},

sandtomb: {

num: 328,

accuracy: 85,

basePower: 35,

category: "Physical",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Sand Tomb",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Ground",

contestType: "Clever",

},

sappyseed: {

num: 738,

accuracy: 90,

basePower: 100,

category: "Physical",

desc: "This move summons Leech Seed on the foe.",

shortDesc: "Summons Leech Seed.",

isNonstandard: "LGPE",

name: "Sappy Seed",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1},

onHit(target, source) {

if (target.hasType('Grass')) return null;

target.addVolatile('leechseed', source);

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Clever",

},

savagespinout: {

num: 634,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Savage Spin-Out",

pp: 1,

priority: 0,

flags: {},

isZ: "buginiumz",

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cool",

},

scald: {

num: 503,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 30% chance to burn the target. The target thaws out if it is frozen.",

shortDesc: "30% chance to burn the target. Thaws target.",

name: "Scald",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

thawsTarget: true,

secondary: {

chance: 30,

status: 'brn',

},

target: "normal",

type: "Water",

contestType: "Tough",

},

scaleshot: {

num: 799,

accuracy: 90,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Lowers the user's Defense by 1 stage and raises the user's Speed by 1 stage after the last hit. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times. User: -1 Def, +1 Spe after last hit.",

name: "Scale Shot",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

selfBoost: {

boosts: {

def: -1,

spe: 1,

},

},

secondary: null,

target: "normal",

type: "Dragon",

zMove: {basePower: 140},

maxMove: {basePower: 130},

},

scaryface: {

num: 184,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Speed by 2 stages.",

shortDesc: "Lowers the target's Speed by 2.",

name: "Scary Face",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

boosts: {

spe: -2,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Tough",

},

scorchingsands: {

num: 815,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "Has a 30% chance to burn the target. The target thaws out if it is frozen.",

shortDesc: "30% chance to burn the target. Thaws target.",

name: "Scorching Sands",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

thawsTarget: true,

secondary: {

chance: 30,

status: 'brn',

},

target: "normal",

type: "Ground",

},

scratch: {

num: 10,

accuracy: 100,

basePower: 40,

category: "Physical",

shortDesc: "No additional effect.",

name: "Scratch",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

screech: {

num: 103,

accuracy: 85,

basePower: 0,

category: "Status",

desc: "Lowers the target's Defense by 2 stages.",

shortDesc: "Lowers the target's Defense by 2.",

name: "Screech",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1, mystery: 1},

boosts: {

def: -2,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Clever",

},

searingshot: {

num: 545,

accuracy: 100,

basePower: 100,

category: "Special",

desc: "Has a 30% chance to burn the target.",

shortDesc: "30% chance to burn adjacent Pokemon.",

isNonstandard: "Past",

name: "Searing Shot",

pp: 5,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'brn',

},

target: "allAdjacent",

type: "Fire",

contestType: "Cool",

},

searingsunrazesmash: {

num: 724,

accuracy: true,

basePower: 200,

category: "Physical",

desc: "This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Ignores the Abilities of other Pokemon.",

isNonstandard: "Past",

name: "Searing Sunraze Smash",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "solganiumz",

ignoreAbility: true,

secondary: null,

target: "normal",

type: "Steel",

contestType: "Cool",

},

secretpower: {

num: 290,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a 30% chance to cause a secondary effect on the target based on the battle terrain. Causes paralysis on the regular Wi-Fi terrain, causes paralysis during Electric Terrain, lowers Special Attack by 1 stage during Misty Terrain, causes sleep during Grassy Terrain and lowers Speed by 1 stage during Psychic Terrain.",

shortDesc: "Effect varies with terrain. (30% paralysis chance)",

isNonstandard: "Past",

name: "Secret Power",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move, pokemon) {

if (this.field.isTerrain('')) return;

move.secondaries = [];

if (this.field.isTerrain('electricterrain')) {

move.secondaries.push({

chance: 30,

status: 'par',

});

} else if (this.field.isTerrain('grassyterrain')) {

move.secondaries.push({

chance: 30,

status: 'slp',

});

} else if (this.field.isTerrain('mistyterrain')) {

move.secondaries.push({

chance: 30,

boosts: {

spa: -1,

},

});

} else if (this.field.isTerrain('psychicterrain')) {

move.secondaries.push({

chance: 30,

boosts: {

spe: -1,

},

});

}

},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Normal",

contestType: "Clever",

},

secretsword: {

num: 548,

accuracy: 100,

basePower: 85,

category: "Special",

defensiveCategory: "Physical",

desc: "Deals damage to the target based on its Defense instead of Special Defense.",

shortDesc: "Damages target based on Defense, not Sp. Def.",

name: "Secret Sword",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Beautiful",

},

seedbomb: {

num: 402,

accuracy: 100,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Seed Bomb",

pp: 15,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Tough",

},

seedflare: {

num: 465,

accuracy: 85,

basePower: 120,

category: "Special",

desc: "Has a 40% chance to lower the target's Special Defense by 2 stages.",

shortDesc: "40% chance to lower the target's Sp. Def by 2.",

isNonstandard: "Past",

name: "Seed Flare",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 40,

boosts: {

spd: -2,

},

},

target: "normal",

type: "Grass",

contestType: "Beautiful",

},

seismictoss: {

num: 69,

accuracy: 100,

basePower: 0,

damage: 'level',

category: "Physical",

desc: "Deals damage to the target equal to the user's level.",

shortDesc: "Does damage equal to the user's level.",

name: "Seismic Toss",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

secondary: null,

target: "normal",

type: "Fighting",

maxMove: {basePower: 75},

contestType: "Tough",

},

selfdestruct: {

num: 120,

accuracy: 100,

basePower: 200,

category: "Physical",

desc: "The user faints after using this move, even if this move fails for having no target. This move is prevented from executing if any active Pokemon has the Damp Ability.",

shortDesc: "Hits adjacent Pokemon. The user faints.",

name: "Self-Destruct",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

selfdestruct: "always",

secondary: null,

target: "allAdjacent",

type: "Normal",

contestType: "Beautiful",

},

shadowball: {

num: 247,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 20% chance to lower the target's Special Defense by 1 stage.",

shortDesc: "20% chance to lower the target's Sp. Def by 1.",

name: "Shadow Ball",

pp: 15,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

boosts: {

spd: -1,

},

},

target: "normal",

type: "Ghost",

contestType: "Clever",

},

shadowbone: {

num: 708,

accuracy: 100,

basePower: 85,

category: "Physical",

desc: "Has a 20% chance to lower the target's Defense by 1 stage.",

shortDesc: "20% chance to lower the target's Defense by 1.",

name: "Shadow Bone",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

boosts: {

def: -1,

},

},

target: "normal",

type: "Ghost",

contestType: "Cool",

},

shadowclaw: {

num: 421,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Shadow Claw",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

shadowforce: {

num: 467,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "If this move is successful, it breaks through the target's Baneful Bunker, Detect, King's Shield, Protect, or Spiky Shield for this turn, allowing other Pokemon to attack the target normally. If the target's side is protected by Crafty Shield, Mat Block, Quick Guard, or Wide Guard, that protection is also broken for this turn and other Pokemon may attack the target's side normally. This attack charges on the first turn and executes on the second. On the first turn, the user avoids all attacks. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Disappears turn 1. Hits turn 2. Breaks protection.",

isNonstandard: "Past",

name: "Shadow Force",

pp: 5,

priority: 0,

flags: {contact: 1, charge: 1, mirror: 1},

breaksProtect: true,

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

condition: {

duration: 2,

onInvulnerability: false,

},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

shadowpunch: {

num: 325,

accuracy: true,

basePower: 60,

category: "Physical",

shortDesc: "This move does not check accuracy.",

name: "Shadow Punch",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Clever",

},

shadowsneak: {

num: 425,

accuracy: 100,

basePower: 40,

category: "Physical",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Shadow Sneak",

pp: 30,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Clever",

},

shadowstrike: {

num: 0,

accuracy: 95,

basePower: 80,

category: "Physical",

desc: "Has a 50% chance to lower the target's Defense by 1 stage.",

shortDesc: "50% chance to lower the target's Defense by 1.",

isNonstandard: "CAP",

name: "Shadow Strike",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 50,

boosts: {

def: -1,

},

},

target: "normal",

type: "Ghost",

contestType: "Clever",

},

sharpen: {

num: 159,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack by 1 stage.",

shortDesc: "Raises the user's Attack by 1.",

isNonstandard: "Past",

name: "Sharpen",

pp: 30,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Cute",

},

shatteredpsyche: {

num: 648,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Shattered Psyche",

pp: 1,

priority: 0,

flags: {},

isZ: "psychiumz",

secondary: null,

target: "normal",

type: "Psychic",

contestType: "Cool",

},

sheercold: {

num: 329,

accuracy: 30,

basePower: 0,

category: "Special",

desc: "Deals damage to the target equal to the target's maximum HP. Ignores accuracy and evasiveness modifiers. This attack's accuracy is equal to (user's level - target's level + X)%, where X is 30 if the user is an Ice type and 20 otherwise, and fails if the target is at a higher level. Ice-type Pokemon and Pokemon with the Sturdy Ability are immune.",

shortDesc: "OHKOs non-Ice targets. Fails if user's lower level.",

name: "Sheer Cold",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

ohko: 'Ice',

target: "normal",

type: "Ice",

zMove: {basePower: 180},

maxMove: {basePower: 130},

contestType: "Beautiful",

},

shellsidearm: {

num: 801,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 20% chance to poison the target. This move becomes a physical attack that makes contact if the value of ((((2 \* the user's level / 5 + 2) \* 90 \* X) / Y) / 50), where X is the user's Attack stat and Y is the target's Defense stat, is greater than the same value where X is the user's Special Attack stat and Y is the target's Special Defense stat. No stat modifiers other than stat stage changes are considered for this purpose. If the two values are equal, this move chooses a damage category at random.",

shortDesc: "20% poison. Phys+contact if it would be stronger.",

name: "Shell Side Arm",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move, pokemon, target) {

const atk = pokemon.getStat('atk', false, true);

const spa = pokemon.getStat('spa', false, true);

const def = target.getStat('def', false, true);

const spd = target.getStat('spd', false, true);

const physical = Math.floor(Math.floor(Math.floor(Math.floor(2 \* pokemon.level / 5 + 2) \* 90 \* atk) / def) / 50);

const special = Math.floor(Math.floor(Math.floor(Math.floor(2 \* pokemon.level / 5 + 2) \* 90 \* spa) / spd) / 50);

if (physical > special || (physical === special && this.random(2) === 0)) {

move.category = 'Physical';

move.flags.contact = 1;

}

},

onHit(target, source, move) {

this.hint(move.category + " Shell Side Arm");

},

onAfterSubDamage(damage, target, source, move) {

this.hint(move.category + " Shell Side Arm");

},

secondary: {

chance: 20,

status: 'psn',

},

target: "normal",

type: "Poison",

},

shellsmash: {

num: 504,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Lowers the user's Defense and Special Defense by 1 stage. Raises the user's Attack, Special Attack, and Speed by 2 stages.",

shortDesc: "Lowers Def, SpD by 1; raises Atk, SpA, Spe by 2.",

name: "Shell Smash",

pp: 15,

priority: 0,

flags: {snatch: 1},

boosts: {

def: -1,

spd: -1,

atk: 2,

spa: 2,

spe: 2,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

shelltrap: {

num: 704,

accuracy: 100,

basePower: 150,

category: "Special",

desc: "Fails unless the user is hit by a physical attack from an opponent this turn before it can execute the move. If the user was hit and has not fainted, it attacks immediately after being hit, and the effect ends. If the opponent's physical attack had a secondary effect removed by the Sheer Force Ability, it does not count for the purposes of this effect.",

shortDesc: "User must take physical damage before moving.",

name: "Shell Trap",

pp: 5,

priority: -3,

flags: {protect: 1},

beforeTurnCallback(pokemon) {

pokemon.addVolatile('shelltrap');

},

onTryMove(pokemon) {

if (!pokemon.volatiles['shelltrap'] || !pokemon.volatiles['shelltrap'].gotHit) {

this.attrLastMove('[still]');

this.add('cant', pokemon, 'Shell Trap', 'Shell Trap');

return null;

}

},

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'move: Shell Trap');

},

onHit(pokemon, source, move) {

if (pokemon.side !== source.side && move.category === 'Physical') {

pokemon.volatiles['shelltrap'].gotHit = true;

const action = this.queue.willMove(pokemon);

if (action) {

this.queue.prioritizeAction(action);

}

}

},

},

secondary: null,

target: "allAdjacentFoes",

type: "Fire",

contestType: "Tough",

},

shiftgear: {

num: 508,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Speed by 2 stages and its Attack by 1 stage.",

shortDesc: "Raises the user's Speed by 2 and Attack by 1.",

name: "Shift Gear",

pp: 10,

priority: 0,

flags: {snatch: 1},

boosts: {

spe: 2,

atk: 1,

},

secondary: null,

target: "self",

type: "Steel",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

shockwave: {

num: 351,

accuracy: true,

basePower: 60,

category: "Special",

shortDesc: "This move does not check accuracy.",

name: "Shock Wave",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

shoreup: {

num: 659,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half down. If the weather is Sandstorm, the user instead restores 2/3 of its maximum HP, rounded half down.",

shortDesc: "User restores 1/2 its max HP; 2/3 in Sandstorm.",

name: "Shore Up",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

onHit(pokemon) {

let factor = 0.5;

if (this.field.isWeather('sandstorm')) {

factor = 0.667;

}

return !!this.heal(this.modify(pokemon.maxhp, factor));

},

secondary: null,

target: "self",

type: "Ground",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

signalbeam: {

num: 324,

accuracy: 100,

basePower: 75,

category: "Special",

desc: "Has a 10% chance to confuse the target.",

shortDesc: "10% chance to confuse the target.",

isNonstandard: "Past",

name: "Signal Beam",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

volatileStatus: 'confusion',

},

target: "normal",

type: "Bug",

contestType: "Beautiful",

},

silverwind: {

num: 318,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 10% chance to raise the user's Attack, Defense, Special Attack, Special Defense, and Speed by 1 stage.",

shortDesc: "10% chance to raise all stats by 1 (not acc/eva).",

isNonstandard: "Past",

name: "Silver Wind",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

self: {

boosts: {

atk: 1,

def: 1,

spa: 1,

spd: 1,

spe: 1,

},

},

},

target: "normal",

type: "Bug",

contestType: "Beautiful",

},

simplebeam: {

num: 493,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target's Ability to become Simple. Fails if the target's Ability is Battle Bond, Comatose, Disguise, Multitype, Power Construct, RKS System, Schooling, Shields Down, Simple, Stance Change, Truant, or Zen Mode.",

shortDesc: "The target's Ability becomes Simple.",

name: "Simple Beam",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onTryHit(pokemon) {

const bannedAbilities = [

'battlebond', 'comatose', 'disguise', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'simple', 'stancechange', 'truant', 'zenmode',

];

if (bannedAbilities.includes(pokemon.ability)) {

return false;

}

},

onHit(pokemon) {

const oldAbility = pokemon.setAbility('simple');

if (oldAbility) {

this.add('-ability', pokemon, 'Simple', '[from] move: Simple Beam');

return;

}

return false;

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spa: 1}},

contestType: "Cute",

},

sing: {

num: 47,

accuracy: 55,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

name: "Sing",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Cute",

},

sinisterarrowraid: {

num: 695,

accuracy: true,

basePower: 180,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Sinister Arrow Raid",

pp: 1,

priority: 0,

flags: {},

isZ: "decidiumz",

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

sizzlyslide: {

num: 735,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "Has a 100% chance to burn the foe.",

shortDesc: "100% chance to burn the foe.",

isNonstandard: "LGPE",

name: "Sizzly Slide",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, defrost: 1},

secondary: {

chance: 100,

status: 'brn',

},

target: "normal",

type: "Fire",

contestType: "Clever",

},

sketch: {

num: 166,

accuracy: true,

basePower: 0,

category: "Status",

desc: "This move is permanently replaced by the last move used by the target. The copied move has the maximum PP for that move. Fails if the target has not made a move, if the user has Transformed, or if the move is Chatter, Sketch, Struggle, or any move the user knows.",

shortDesc: "Permanently copies the last move target used.",

isNonstandard: "Past",

name: "Sketch",

pp: 1,

noPPBoosts: true,

priority: 0,

flags: {authentic: 1, mystery: 1},

onHit(target, source) {

const disallowedMoves = ['chatter', 'sketch', 'struggle'];

const move = target.lastMove;

if (source.transformed || !move || source.moves.includes(move.id)) return false;

if (disallowedMoves.includes(move.id) || move.isZ || move.isMax) return false;

const sketchIndex = source.moves.indexOf('sketch');

if (sketchIndex < 0) return false;

const sketchedMove = {

move: move.name,

id: move.id,

pp: move.pp,

maxpp: move.pp,

target: move.target,

disabled: false,

used: false,

};

source.moveSlots[sketchIndex] = sketchedMove;

source.baseMoveSlots[sketchIndex] = sketchedMove;

this.add('-activate', source, 'move: Sketch', move.name);

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Clever",

},

skillswap: {

num: 285,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps its Ability with the target's Ability. Fails if either the user or the target's Ability is Battle Bond, Comatose, Disguise, Gulp Missile, Hunger Switch, Ice Face, Illusion, Multitype, Neutralizing Gas, Power Construct, RKS System, Schooling, Shields Down, Stance Change, Wonder Guard, or Zen Mode.",

shortDesc: "The user and the target trade Abilities.",

name: "Skill Swap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, authentic: 1, mystery: 1},

onTryHit(target, source) {

const bannedAbilities = [

'battlebond', 'comatose', 'disguise', 'gulpmissile', 'hungerswitch', 'iceface', 'illusion', 'multitype', 'neutralizinggas', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'wonderguard', 'zenmode',

];

if (

target.volatiles['dynamax'] || bannedAbilities.includes(target.ability) || bannedAbilities.includes(source.ability)

) {

return false;

}

},

onHit(target, source, move) {

const targetAbility = target.getAbility();

const sourceAbility = source.getAbility();

if (target.side === source.side) {

this.add('-activate', source, 'move: Skill Swap', '', '', '[of] ' + target);

} else {

this.add('-activate', source, 'move: Skill Swap', targetAbility, sourceAbility, '[of] ' + target);

}

this.singleEvent('End', sourceAbility, source.abilityData, source);

this.singleEvent('End', targetAbility, target.abilityData, target);

if (targetAbility.id !== sourceAbility.id) {

source.ability = targetAbility.id;

target.ability = sourceAbility.id;

source.abilityData = {id: toID(source.ability), target: source};

target.abilityData = {id: toID(target.ability), target: target};

if (target.side !== source.side) target.staleness = 'external';

}

this.singleEvent('Start', targetAbility, source.abilityData, source);

this.singleEvent('Start', sourceAbility, target.abilityData, target);

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

skittersmack: {

num: 806,

accuracy: 90,

basePower: 70,

category: "Physical",

desc: "Has a 100% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "100% chance to lower target's Sp. Atk by 1.",

name: "Skitter Smack",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spa: -1,

},

},

target: "normal",

type: "Bug",

},

skullbash: {

num: 130,

accuracy: 100,

basePower: 130,

category: "Physical",

desc: "This attack charges on the first turn and executes on the second. Raises the user's Defense by 1 stage on the first turn. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Raises user's Defense by 1 on turn 1. Hits turn 2.",

name: "Skull Bash",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

this.boost({def: 1}, attacker, attacker, move);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

skyattack: {

num: 143,

accuracy: 90,

basePower: 140,

category: "Physical",

desc: "Has a 30% chance to flinch the target and a higher chance for a critical hit. This attack charges on the first turn and executes on the second. If the user is holding a Power Herb, the move completes in one turn.",

shortDesc: "Charges, then hits turn 2. 30% flinch. High crit.",

name: "Sky Attack",

pp: 5,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1, distance: 1},

critRatio: 2,

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "any",

type: "Flying",

contestType: "Cool",

},

skydrop: {

num: 507,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "This attack takes the target into the air with the user on the first turn and executes on the second. Pokemon weighing 200 kg or more cannot be lifted. On the first turn, the user and the target avoid all attacks other than Gust, Hurricane, Sky Uppercut, Smack Down, Thousand Arrows, Thunder, and Twister. The user and the target cannot make a move between turns, but the target can select a move to use. This move cannot damage Flying-type Pokemon. Fails on the first turn if the target is an ally, if the target has a substitute, or if the target is using Bounce, Dig, Dive, Fly, Phantom Force, Shadow Force, or Sky Drop.",

shortDesc: "User and foe fly up turn 1. Damages on turn 2.",

isNonstandard: "Past",

name: "Sky Drop",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1, gravity: 1, distance: 1},

onModifyMove(move, source) {

if (!source.volatiles['skydrop']) {

move.accuracy = true;

move.flags.contact = 0;

}

},

onMoveFail(target, source) {

if (source.volatiles['twoturnmove'] && source.volatiles['twoturnmove'].duration === 1) {

source.removeVolatile('skydrop');

source.removeVolatile('twoturnmove');

this.add('-end', target, 'Sky Drop', '[interrupt]');

}

},

onTryHit(target, source, move) {

if (target.fainted) return false;

if (source.removeVolatile(move.id)) {

if (target !== source.volatiles['twoturnmove'].source) return false;

if (target.hasType('Flying')) {

this.add('-immune', target);

return null;

}

} else {

if (target.volatiles['substitute'] || target.side === source.side) {

return false;

}

if (target.getWeight() >= 2000) {

this.add('-fail', target, 'move: Sky Drop', '[heavy]');

return null;

}

this.add('-prepare', source, move.name, target);

source.addVolatile('twoturnmove', target);

return null;

}

},

onHit(target, source) {

if (target.hp) this.add('-end', target, 'Sky Drop');

},

condition: {

duration: 2,

onAnyDragOut(pokemon) {

if (pokemon === this.effectData.target || pokemon === this.effectData.source) return false;

},

onFoeTrapPokemonPriority: -15,

onFoeTrapPokemon(defender) {

if (defender !== this.effectData.source) return;

defender.trapped = true;

},

onFoeBeforeMovePriority: 12,

onFoeBeforeMove(attacker, defender, move) {

if (attacker === this.effectData.source) {

attacker.activeMoveActions--;

this.debug('Sky drop nullifying.');

return null;

}

},

onRedirectTargetPriority: 99,

onRedirectTarget(target, source, source2) {

if (source !== this.effectData.target) return;

if (this.effectData.source.fainted) return;

return this.effectData.source;

},

onAnyInvulnerability(target, source, move) {

if (target !== this.effectData.target && target !== this.effectData.source) {

return;

}

if (source === this.effectData.target && target === this.effectData.source) {

return;

}

if (['gust', 'twister', 'skyuppercut', 'thunder', 'hurricane', 'smackdown', 'thousandarrows'].includes(move.id)) {

return;

}

return false;

},

onAnyBasePower(basePower, target, source, move) {

if (target !== this.effectData.target && target !== this.effectData.source) {

return;

}

if (source === this.effectData.target && target === this.effectData.source) {

return;

}

if (move.id === 'gust' || move.id === 'twister') {

return this.chainModify(2);

}

},

onFaint(target) {

if (target.volatiles['skydrop'] && target.volatiles['twoturnmove'].source) {

this.add('-end', target.volatiles['twoturnmove'].source, 'Sky Drop', '[interrupt]');

}

},

},

secondary: null,

target: "any",

type: "Flying",

contestType: "Tough",

},

skyuppercut: {

num: 327,

accuracy: 90,

basePower: 85,

category: "Physical",

desc: "This move can hit a target using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop.",

shortDesc: "Can hit Pokemon using Bounce, Fly, or Sky Drop.",

isNonstandard: "Past",

name: "Sky Uppercut",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

slackoff: {

num: 303,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Heals the user by 50% of its max HP.",

name: "Slack Off",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

slam: {

num: 21,

accuracy: 75,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Slam",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

slash: {

num: 163,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Slash",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

sleeppowder: {

num: 79,

accuracy: 75,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

name: "Sleep Powder",

pp: 15,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

sleeptalk: {

num: 214,

accuracy: true,

basePower: 0,

category: "Status",

desc: "One of the user's known moves, besides this move, is selected for use at random. Fails if the user is not asleep. The selected move does not have PP deducted from it, and can currently have 0 PP. This move cannot select Assist, Beak Blast, Belch, Bide, Celebrate, Chatter, Copycat, Dynamax Cannon, Focus Punch, Hold Hands, Me First, Metronome, Mimic, Mirror Move, Nature Power, Shell Trap, Sketch, Sleep Talk, Struggle, Uproar, any two-turn move, or any Max Move.",

shortDesc: "User must be asleep. Uses another known move.",

name: "Sleep Talk",

pp: 10,

priority: 0,

flags: {},

sleepUsable: true,

onTryHit(pokemon) {

if (pokemon.status !== 'slp' && !pokemon.hasAbility('comatose')) return false;

},

onHit(pokemon) {

const noSleepTalk = [

'assist', 'beakblast', 'belch', 'bide', 'celebrate', 'chatter', 'copycat', 'dynamaxcannon', 'focuspunch', 'mefirst', 'metronome', 'mimic', 'mirrormove', 'naturepower', 'shelltrap', 'sketch', 'sleeptalk', 'uproar',

];

const moves = [];

for (const moveSlot of pokemon.moveSlots) {

const moveid = moveSlot.id;

if (!moveid) continue;

const move = this.dex.getMove(moveid);

if (noSleepTalk.includes(moveid) || move.flags['charge'] || (move.isZ && move.basePower !== 1)) {

continue;

}

moves.push(moveid);

}

let randomMove = '';

if (moves.length) randomMove = this.sample(moves);

if (!randomMove) {

return false;

}

this.useMove(randomMove, pokemon);

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'crit2'},

contestType: "Cute",

},

sludge: {

num: 124,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Has a 30% chance to poison the target.",

shortDesc: "30% chance to poison the target.",

name: "Sludge",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Tough",

},

sludgebomb: {

num: 188,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 30% chance to poison the target.",

shortDesc: "30% chance to poison the target.",

name: "Sludge Bomb",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Tough",

},

sludgewave: {

num: 482,

accuracy: 100,

basePower: 95,

category: "Special",

desc: "Has a 10% chance to poison the target.",

shortDesc: "10% chance to poison adjacent Pokemon.",

name: "Sludge Wave",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'psn',

},

target: "allAdjacent",

type: "Poison",

contestType: "Tough",

},

smackdown: {

num: 479,

accuracy: 100,

basePower: 50,

category: "Physical",

desc: "This move can hit a target using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop. If this move hits a target under the effect of Bounce, Fly, Magnet Rise, or Telekinesis, the effect ends. If the target is a Flying type that has not used Roost this turn or a Pokemon with the Levitate Ability, it loses its immunity to Ground-type attacks and the Arena Trap Ability as long as it remains active. During the effect, Magnet Rise fails for the target and Telekinesis fails against the target.",

shortDesc: "Removes the target's Ground immunity.",

name: "Smack Down",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

volatileStatus: 'smackdown',

condition: {

noCopy: true,

onStart(pokemon) {

let applies = false;

if (pokemon.hasType('Flying') || pokemon.hasAbility('levitate')) applies = true;

if (pokemon.hasItem('ironball') || pokemon.volatiles['ingrain'] ||

this.field.getPseudoWeather('gravity')) applies = false;

if (pokemon.removeVolatile('fly') || pokemon.removeVolatile('bounce')) {

applies = true;

this.queue.cancelMove(pokemon);

pokemon.removeVolatile('twoturnmove');

}

if (pokemon.volatiles['magnetrise']) {

applies = true;

delete pokemon.volatiles['magnetrise'];

}

if (pokemon.volatiles['telekinesis']) {

applies = true;

delete pokemon.volatiles['telekinesis'];

}

if (!applies) return false;

this.add('-start', pokemon, 'Smack Down');

},

onRestart(pokemon) {

if (pokemon.removeVolatile('fly') || pokemon.removeVolatile('bounce')) {

this.queue.cancelMove(pokemon);

this.add('-start', pokemon, 'Smack Down');

}

},

// groundedness implemented in battle.engine.js:BattlePokemon#isGrounded

},

secondary: null,

target: "normal",

type: "Rock",

contestType: "Tough",

},

smartstrike: {

num: 684,

accuracy: true,

basePower: 70,

category: "Physical",

shortDesc: "This move does not check accuracy.",

name: "Smart Strike",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Steel",

contestType: "Cool",

},

smellingsalts: {

num: 265,

accuracy: 100,

basePower: 70,

basePowerCallback(pokemon, target, move) {

if (target.status === 'par') return move.basePower \* 2;

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the target is paralyzed. If the user has not fainted, the target is cured of paralysis.",

shortDesc: "Power doubles if target is paralyzed, and cures it.",

isNonstandard: "Past",

name: "Smelling Salts",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onHit(target) {

if (target.status === 'par') target.cureStatus();

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

smog: {

num: 123,

accuracy: 70,

basePower: 30,

category: "Special",

desc: "Has a 40% chance to poison the target.",

shortDesc: "40% chance to poison the target.",

name: "Smog",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 40,

status: 'psn',

},

target: "normal",

type: "Poison",

contestType: "Tough",

},

smokescreen: {

num: 108,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's accuracy by 1 stage.",

shortDesc: "Lowers the target's accuracy by 1.",

name: "Smokescreen",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

accuracy: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {evasion: 1}},

contestType: "Clever",

},

snaptrap: {

num: 779,

accuracy: 100,

basePower: 35,

category: "Physical",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Snap Trap",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Grass",

},

snarl: {

num: 555,

accuracy: 95,

basePower: 55,

category: "Special",

desc: "Has a 100% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Sp. Atk by 1.",

name: "Snarl",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: {

chance: 100,

boosts: {

spa: -1,

},

},

target: "allAdjacentFoes",

type: "Dark",

contestType: "Tough",

},

snatch: {

num: 289,

accuracy: true,

basePower: 0,

category: "Status",

desc: "If another Pokemon uses certain non-damaging moves this turn, the user steals that move to use itself. If multiple Pokemon use one of those moves this turn, the applicable moves are all stolen by the first Pokemon in turn order that used this move this turn. This effect is ignored while the user is under the effect of Sky Drop.",

shortDesc: "User steals certain support moves to use itself.",

isNonstandard: "Past",

name: "Snatch",

pp: 10,

priority: 4,

flags: {authentic: 1},

volatileStatus: 'snatch',

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'Snatch');

},

onAnyTryMove(source, target, move) {

const snatchUser = this.effectData.source;

if (snatchUser.isSkyDropped()) return;

if (!move || move.isZ || move.isMax || !move.flags['snatch'] || move.sourceEffect === 'snatch') {

return;

}

snatchUser.removeVolatile('snatch');

this.add('-activate', snatchUser, 'move: Snatch', '[of] ' + source);

this.useMove(move.id, snatchUser);

return null;

},

},

secondary: null,

pressureTarget: "foeSide",

target: "self",

type: "Dark",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

snipeshot: {

num: 745,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a higher chance for a critical hit. This move cannot be redirected to a different target by any effect.",

shortDesc: "High critical hit ratio. Cannot be redirected.",

name: "Snipe Shot",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

tracksTarget: true,

secondary: null,

target: "normal",

type: "Water",

},

snore: {

num: 173,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "Has a 30% chance to flinch the target. Fails if the user is not asleep.",

shortDesc: "User must be asleep. 30% chance to flinch target.",

name: "Snore",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

sleepUsable: true,

onTryHit(target, source) {

if (source.status !== 'slp' && !source.hasAbility('comatose')) return false;

},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Normal",

contestType: "Cute",

},

soak: {

num: 487,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to become a Water type. Fails if the target is an Arceus or a Silvally, or if the target is already purely Water type.",

shortDesc: "Changes the target's type to Water.",

name: "Soak",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target) {

if (target.getTypes().join() === 'Water' || !target.setType('Water')) {

// Soak should animate even when it fails.

// Returning false would suppress the animation.

this.add('-fail', target);

return null;

}

this.add('-start', target, 'typechange', 'Water');

},

secondary: null,

target: "normal",

type: "Water",

zMove: {boost: {spa: 1}},

contestType: "Cute",

},

softboiled: {

num: 135,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP, rounded half up.",

shortDesc: "Heals the user by 50% of its max HP.",

name: "Soft-Boiled",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

heal: [1, 2],

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

solarbeam: {

num: 76,

accuracy: 100,

basePower: 120,

category: "Special",

desc: "This attack charges on the first turn and executes on the second. Power is halved if the weather is Hail, Primordial Sea, Rain Dance, or Sandstorm and the user is not holding Utility Umbrella. If the user is holding a Power Herb or the weather is Desolate Land or Sunny Day, the move completes in one turn. If the user is holding Utility Umbrella and the weather is Desolate Land or Sunny Day, the move still requires a turn to charge.",

shortDesc: "Charges turn 1. Hits turn 2. No charge in sunlight.",

name: "Solar Beam",

pp: 10,

priority: 0,

flags: {charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (['sunnyday', 'desolateland'].includes(attacker.effectiveWeather())) {

this.attrLastMove('[still]');

this.addMove('-anim', attacker, move.name, defender);

return;

}

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

onBasePower(basePower, pokemon, target) {

if (['raindance', 'primordialsea', 'sandstorm', 'hail'].includes(pokemon.effectiveWeather())) {

this.debug('weakened by weather');

return this.chainModify(0.5);

}

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

solarblade: {

num: 669,

accuracy: 100,

basePower: 125,

category: "Physical",

desc: "This attack charges on the first turn and executes on the second. Power is halved if the weather is Hail, Primordial Sea, Rain Dance, or Sandstorm and the user is not holding Utility Umbrella. If the user is holding a Power Herb or the weather is Desolate Land or Sunny Day, the move completes in one turn. If the user is holding Utility Umbrella and the weather is Desolate Land or Sunny Day, the move still requires a turn to charge.",

shortDesc: "Charges turn 1. Hits turn 2. No charge in sunlight.",

name: "Solar Blade",

pp: 10,

priority: 0,

flags: {contact: 1, charge: 1, protect: 1, mirror: 1},

onTryMove(attacker, defender, move) {

if (attacker.removeVolatile(move.id)) {

return;

}

this.add('-prepare', attacker, move.name, defender);

if (['sunnyday', 'desolateland'].includes(attacker.effectiveWeather())) {

this.attrLastMove('[still]');

this.addMove('-anim', attacker, move.name, defender);

return;

}

if (!this.runEvent('ChargeMove', attacker, defender, move)) {

return;

}

attacker.addVolatile('twoturnmove', defender);

return null;

},

onBasePower(basePower, pokemon, target) {

if (['raindance', 'primordialsea', 'sandstorm', 'hail'].includes(pokemon.effectiveWeather())) {

this.debug('weakened by weather');

return this.chainModify(0.5);

}

},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

sonicboom: {

num: 49,

accuracy: 90,

basePower: 0,

damage: 20,

category: "Special",

desc: "Deals 20 HP of damage to the target.",

shortDesc: "Always does 20 HP of damage.",

isNonstandard: "Past",

name: "Sonic Boom",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

soulstealing7starstrike: {

num: 699,

accuracy: true,

basePower: 195,

category: "Physical",

shortDesc: "No additional effect.",

isNonstandard: "Past",

name: "Soul-Stealing 7-Star Strike",

pp: 1,

priority: 0,

flags: {contact: 1},

isZ: "marshadiumz",

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

spacialrend: {

num: 460,

accuracy: 95,

basePower: 100,

category: "Special",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

isNonstandard: "Past",

name: "Spacial Rend",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Dragon",

contestType: "Beautiful",

},

spark: {

num: 209,

accuracy: 100,

basePower: 65,

category: "Physical",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze the target.",

name: "Spark",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

sparklingaria: {

num: 664,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "If the user has not fainted, the target is cured of its burn.",

shortDesc: "The target is cured of its burn.",

name: "Sparkling Aria",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

secondary: {

dustproof: true,

chance: 100,

onHit(target) {

if (target.status === 'brn') target.cureStatus();

},

},

target: "allAdjacent",

type: "Water",

contestType: "Tough",

},

sparklyswirl: {

num: 740,

accuracy: 85,

basePower: 120,

category: "Special",

desc: "Every Pokemon in the user's party is cured of its major status condition.",

shortDesc: "Cures the user's party of all status conditions.",

isNonstandard: "LGPE",

name: "Sparkly Swirl",

pp: 5,

priority: 0,

flags: {protect: 1},

self: {

onHit(pokemon, source, move) {

this.add('-activate', source, 'move: Aromatherapy');

for (const ally of source.side.pokemon) {

if (ally !== source && (ally.volatiles['substitute'] && !move.infiltrates)) {

continue;

}

ally.cureStatus();

}

},

},

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Clever",

},

spectralthief: {

num: 712,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "The target's stat stages greater than 0 are stolen from it and applied to the user before dealing damage.",

shortDesc: "Steals target's boosts before dealing damage.",

name: "Spectral Thief",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, authentic: 1},

stealsBoosts: true,

// Boost stealing implemented in scripts.js

secondary: null,

target: "normal",

type: "Ghost",

contestType: "Cool",

},

speedswap: {

num: 683,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user swaps its Speed stat with the target. Stat stage changes are unaffected.",

shortDesc: "Swaps Speed stat with target.",

name: "Speed Swap",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, authentic: 1, mystery: 1},

onHit(target, source) {

const targetSpe = target.storedStats.spe;

target.storedStats.spe = source.storedStats.spe;

source.storedStats.spe = targetSpe;

this.add('-activate', source, 'move: Speed Swap', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

spiderweb: {

num: 169,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Prevents the target from switching out.",

isNonstandard: "Past",

name: "Spider Web",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

onHit(target, source, move) {

return target.addVolatile('trapped', source, move, 'trapper');

},

secondary: null,

target: "normal",

type: "Bug",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

spikecannon: {

num: 131,

accuracy: 100,

basePower: 20,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

isNonstandard: "Past",

name: "Spike Cannon",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

maxMove: {basePower: 120},

contestType: "Cool",

},

spikes: {

num: 191,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Sets up a hazard on the opposing side of the field, damaging each opposing Pokemon that switches in, unless it is a Flying-type Pokemon or has the Levitate Ability. Can be used up to three times before failing. Opponents lose 1/8 of their maximum HP with one layer, 1/6 of their maximum HP with two layers, and 1/4 of their maximum HP with three layers, all rounded down. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, or is hit by Defog.",

shortDesc: "Hurts grounded foes on switch-in. Max 3 layers.",

name: "Spikes",

pp: 20,

priority: 0,

flags: {reflectable: 1, nonsky: 1},

sideCondition: 'spikes',

condition: {

// this is a side condition

onStart(side) {

this.add('-sidestart', side, 'Spikes');

this.effectData.layers = 1;

},

onRestart(side) {

if (this.effectData.layers >= 3) return false;

this.add('-sidestart', side, 'Spikes');

this.effectData.layers++;

},

onSwitchIn(pokemon) {

if (!pokemon.isGrounded()) return;

if (pokemon.hasItem('heavydutyboots')) return;

const damageAmounts = [0, 3, 4, 6]; // 1/8, 1/6, 1/4

this.damage(damageAmounts[this.effectData.layers] \* pokemon.maxhp / 24);

},

},

secondary: null,

target: "foeSide",

type: "Ground",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

spikyshield: {

num: 596,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user is protected from most attacks made by other Pokemon during this turn, and Pokemon making contact with the user lose 1/8 of their maximum HP, rounded down. This move has a 1/X chance of being successful, where X starts at 1 and triples each time this move is successfully used. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn.",

shortDesc: "Protects from moves. Contact: loses 1/8 max HP.",

name: "Spiky Shield",

pp: 10,

priority: 4,

flags: {},

stallingMove: true,

volatileStatus: 'spikyshield',

onTryHit(target, source, move) {

return !!this.queue.willAct() && this.runEvent('StallMove', target);

},

onHit(pokemon) {

pokemon.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target) {

this.add('-singleturn', target, 'move: Protect');

},

onTryHitPriority: 3,

onTryHit(target, source, move) {

if (!move.flags['protect']) {

if (move.isZ || (move.isMax && !move.breaksProtect)) target.getMoveHitData(move).zBrokeProtect = true;

return;

}

if (move.smartTarget) {

move.smartTarget = false;

} else {

this.add('-activate', target, 'move: Protect');

}

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

if (move.flags['contact']) {

this.damage(source.baseMaxhp / 8, source, target);

}

return this.NOT\_FAIL;

},

onHit(target, source, move) {

if (move.isZOrMaxPowered && move.flags['contact']) {

this.damage(source.baseMaxhp / 8, source, target);

}

},

},

secondary: null,

target: "self",

type: "Grass",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

spiritbreak: {

num: 789,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "Has a 100% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "100% chance to lower the target's Sp. Atk by 1.",

name: "Spirit Break",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spa: -1,

},

},

target: "normal",

type: "Fairy",

},

spiritshackle: {

num: 662,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Prevents the target from switching out.",

name: "Spirit Shackle",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

onHit(target, source, move) {

if (source.isActive) target.addVolatile('trapped', source, move, 'trapper');

},

},

target: "normal",

type: "Ghost",

contestType: "Tough",

},

spitup: {

num: 255,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon) {

if (!pokemon.volatiles['stockpile'] || !pokemon.volatiles['stockpile'].layers) return false;

return pokemon.volatiles['stockpile'].layers \* 100;

},

category: "Special",

desc: "Power is equal to 100 times the user's Stockpile count. Fails if the user's Stockpile count is 0. Whether or not this move is successful, the user's Defense and Special Defense decrease by as many stages as Stockpile had increased them, and the user's Stockpile count resets to 0.",

shortDesc: "More power with more uses of Stockpile.",

name: "Spit Up",

pp: 10,

priority: 0,

flags: {protect: 1},

onTry(pokemon) {

if (!pokemon.volatiles['stockpile']) {

return false;

}

},

onAfterMove(pokemon) {

pokemon.removeVolatile('stockpile');

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

spite: {

num: 180,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target's last move used to lose 4 PP. Fails if the target has not made a move, if the move has 0 PP, or if it no longer knows the move.",

shortDesc: "Lowers the PP of the target's last move by 4.",

name: "Spite",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

onHit(target) {

const move = target.lastMove;

if (!move || move.isZ || move.isMax) return false;

const ppDeducted = target.deductPP(move.id, 4);

if (!ppDeducted) return false;

this.add("-activate", target, 'move: Spite', move.name, ppDeducted);

},

secondary: null,

target: "normal",

type: "Ghost",

zMove: {effect: 'heal'},

contestType: "Tough",

},

splash: {

num: 150,

accuracy: true,

basePower: 0,

category: "Status",

shortDesc: "No competitive use.",

name: "Splash",

pp: 40,

priority: 0,

flags: {gravity: 1},

onTryHit(target, source) {

this.add('-nothing');

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 3}},

contestType: "Cute",

},

splinteredstormshards: {

num: 727,

accuracy: true,

basePower: 190,

category: "Physical",

desc: "Ends the effects of Electric Terrain, Grassy Terrain, Misty Terrain, and Psychic Terrain.",

shortDesc: "Ends the effects of terrain.",

isNonstandard: "Past",

name: "Splintered Stormshards",

pp: 1,

priority: 0,

flags: {},

onHit() {

this.field.clearTerrain();

},

isZ: "lycaniumz",

secondary: null,

target: "normal",

type: "Rock",

contestType: "Cool",

},

splishysplash: {

num: 730,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 30% chance to paralyze the target.",

shortDesc: "30% chance to paralyze the target.",

isNonstandard: "LGPE",

name: "Splishy Splash",

pp: 15,

priority: 0,

flags: {protect: 1},

secondary: {

chance: 30,

status: 'par',

},

target: "allAdjacentFoes",

type: "Water",

contestType: "Cool",

},

spore: {

num: 147,

accuracy: 100,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to fall asleep.",

name: "Spore",

pp: 15,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1},

status: 'slp',

secondary: null,

target: "normal",

type: "Grass",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

spotlight: {

num: 671,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Until the end of the turn, all single-target attacks from opponents of the target are redirected to the target. Such attacks are redirected to the target before they can be reflected by Magic Coat or the Magic Bounce Ability, or drawn in by the Lightning Rod or Storm Drain Abilities. Fails if it is not a Double Battle or Battle Royal.",

shortDesc: "Target's foes' moves are redirected to it this turn.",

name: "Spotlight",

pp: 15,

priority: 3,

flags: {protect: 1, reflectable: 1, mystery: 1},

volatileStatus: 'spotlight',

onTryHit(target) {

if (target.side.active.length < 2) return false;

},

condition: {

duration: 1,

onStart(pokemon) {

this.add('-singleturn', pokemon, 'move: Spotlight');

},

onFoeRedirectTargetPriority: 2,

onFoeRedirectTarget(target, source, source2, move) {

if (this.validTarget(this.effectData.target, source, move.target)) {

this.debug("Spotlight redirected target of move");

return this.effectData.target;

}

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

stealthrock: {

num: 446,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Sets up a hazard on the opposing side of the field, damaging each opposing Pokemon that switches in. Fails if the effect is already active on the opposing side. Foes lose 1/32, 1/16, 1/8, 1/4, or 1/2 of their maximum HP, rounded down, based on their weakness to the Rock type; 0.25x, 0.5x, neutral, 2x, or 4x, respectively. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, or is hit by Defog.",

shortDesc: "Hurts foes on switch-in. Factors Rock weakness.",

name: "Stealth Rock",

pp: 20,

priority: 0,

flags: {reflectable: 1},

sideCondition: 'stealthrock',

condition: {

// this is a side condition

onStart(side) {

this.add('-sidestart', side, 'move: Stealth Rock');

},

onSwitchIn(pokemon) {

if (pokemon.hasItem('heavydutyboots')) return;

const typeMod = this.clampIntRange(pokemon.runEffectiveness(this.dex.getActiveMove('stealthrock')), -6, 6);

this.damage(pokemon.maxhp \* Math.pow(2, typeMod) / 8);

},

},

secondary: null,

target: "foeSide",

type: "Rock",

zMove: {boost: {def: 1}},

contestType: "Cool",

},

steameruption: {

num: 592,

accuracy: 95,

basePower: 110,

category: "Special",

desc: "Has a 30% chance to burn the target. The target thaws out if it is frozen.",

shortDesc: "30% chance to burn the target. Thaws target.",

isNonstandard: "Past",

name: "Steam Eruption",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1, defrost: 1},

thawsTarget: true,

secondary: {

chance: 30,

status: 'brn',

},

target: "normal",

type: "Water",

contestType: "Beautiful",

},

steamroller: {

num: 537,

accuracy: 100,

basePower: 65,

category: "Physical",

desc: "Has a 30% chance to flinch the target. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "30% chance to flinch the target.",

isNonstandard: "Past",

name: "Steamroller",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Bug",

contestType: "Tough",

},

steelbeam: {

num: 796,

accuracy: 95,

basePower: 140,

category: "Special",

desc: "Whether or not this move is successful and even if it would cause fainting, the user loses 1/2 of its maximum HP, rounded up, unless the user has the Magic Guard Ability.",

shortDesc: "User loses 50% max HP.",

name: "Steel Beam",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

mindBlownRecoil: true,

onAfterMove(pokemon, target, move) {

if (move.mindBlownRecoil && !move.multihit) {

this.damage(Math.round(pokemon.maxhp / 2), pokemon, pokemon, this.dex.getEffect('Steel Beam'), true);

}

},

secondary: null,

target: "normal",

type: "Steel",

},

steelroller: {

num: 798,

accuracy: 100,

basePower: 130,

category: "Physical",

desc: "Fails if there is no terrain active. Ends the effects of Electric Terrain, Grassy Terrain, Misty Terrain, and Psychic Terrain.",

shortDesc: "Fails if there is no terrain active. Ends the effects of terrain.",

name: "Steel Roller",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onTryHit() {

if (this.field.isTerrain('')) return false;

},

onHit() {

this.field.clearTerrain();

},

secondary: null,

target: "normal",

type: "Steel",

},

steelwing: {

num: 211,

accuracy: 90,

basePower: 70,

category: "Physical",

desc: "Has a 10% chance to raise the user's Defense by 1 stage.",

shortDesc: "10% chance to raise the user's Defense by 1.",

name: "Steel Wing",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 10,

self: {

boosts: {

def: 1,

},

},

},

target: "normal",

type: "Steel",

contestType: "Cool",

},

stickyweb: {

num: 564,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Sets up a hazard on the opposing side of the field, lowering the Speed by 1 stage of each opposing Pokemon that switches in, unless it is a Flying-type Pokemon or has the Levitate Ability. Fails if the effect is already active on the opposing side. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, or is hit by Defog.",

shortDesc: "Lowers Speed of grounded foes by 1 on switch-in.",

name: "Sticky Web",

pp: 20,

priority: 0,

flags: {reflectable: 1},

sideCondition: 'stickyweb',

condition: {

onStart(side) {

this.add('-sidestart', side, 'move: Sticky Web');

},

onSwitchIn(pokemon) {

if (!pokemon.isGrounded()) return;

if (pokemon.hasItem('heavydutyboots')) return;

this.add('-activate', pokemon, 'move: Sticky Web');

this.boost({spe: -1}, pokemon, this.effectData.source, this.dex.getActiveMove('stickyweb'));

},

},

secondary: null,

target: "foeSide",

type: "Bug",

zMove: {boost: {spe: 1}},

contestType: "Tough",

},

stockpile: {

num: 254,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense and Special Defense by 1 stage. The user's Stockpile count increases by 1. Fails if the user's Stockpile count is 3. The user's Stockpile count is reset to 0 when it is no longer active.",

shortDesc: "Raises user's Defense, Sp. Def by 1. Max 3 uses.",

name: "Stockpile",

pp: 20,

priority: 0,

flags: {snatch: 1},

onTryHit(pokemon) {

if (pokemon.volatiles['stockpile'] && pokemon.volatiles['stockpile'].layers >= 3) return false;

},

volatileStatus: 'stockpile',

condition: {

noCopy: true,

onStart(target) {

this.effectData.layers = 1;

this.effectData.def = 0;

this.effectData.spd = 0;

this.add('-start', target, 'stockpile' + this.effectData.layers);

const [curDef, curSpD] = [target.boosts.def, target.boosts.spd];

this.boost({def: 1, spd: 1}, target, target);

if (curDef !== target.boosts.def) this.effectData.def--;

if (curSpD !== target.boosts.spd) this.effectData.spd--;

},

onRestart(target) {

if (this.effectData.layers >= 3) return false;

this.effectData.layers++;

this.add('-start', target, 'stockpile' + this.effectData.layers);

const curDef = target.boosts.def;

const curSpD = target.boosts.spd;

this.boost({def: 1, spd: 1}, target, target);

if (curDef !== target.boosts.def) this.effectData.def--;

if (curSpD !== target.boosts.spd) this.effectData.spd--;

},

onEnd(target) {

if (this.effectData.def || this.effectData.spd) {

const boosts: SparseBoostsTable = {};

if (this.effectData.def) boosts.def = this.effectData.def;

if (this.effectData.spd) boosts.spd = this.effectData.spd;

this.boost(boosts, target, target);

}

this.add('-end', target, 'Stockpile');

if (this.effectData.def !== this.effectData.layers \* -1 || this.effectData.spd !== this.effectData.layers \* -1) {

this.hint("In Gen 7, Stockpile keeps track of how many times it successfully altered each stat individually.");

}

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Tough",

},

stokedsparksurfer: {

num: 700,

accuracy: true,

basePower: 175,

category: "Special",

desc: "Has a 100% chance to paralyze the target.",

shortDesc: "100% chance to paralyze the target.",

isNonstandard: "Past",

name: "Stoked Sparksurfer",

pp: 1,

priority: 0,

flags: {},

isZ: "aloraichiumz",

secondary: {

chance: 100,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

stomp: {

num: 23,

accuracy: 100,

basePower: 65,

category: "Physical",

desc: "Has a 30% chance to flinch the target. Damage doubles and no accuracy check is done if the target has used Minimize while active.",

shortDesc: "30% chance to flinch the target.",

name: "Stomp",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, nonsky: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Normal",

contestType: "Tough",

},

stompingtantrum: {

num: 707,

accuracy: 100,

basePower: 75,

basePowerCallback(pokemon, target, move) {

if (pokemon.moveLastTurnResult === false) return move.basePower \* 2;

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the user's last move on the previous turn, including moves called by other moves or those used through Instruct, Magic Coat, Snatch, or the Dancer or Magic Bounce Abilities, failed to do any of its normal effects, not including damage from an unsuccessful High Jump Kick, Jump Kick, or Mind Blown, or if the user was prevented from moving by any effect other than recharging or Sky Drop. A move that was blocked by Baneful Bunker, Detect, King's Shield, Protect, Spiky Shield, Crafty Shield, Mat Block, Quick Guard, or Wide Guard will not double this move's power, nor will Bounce or Fly ending early due to the effect of Gravity, Smack Down, or Thousand Arrows.",

shortDesc: "Power doubles if the user's last move failed.",

name: "Stomping Tantrum",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Ground",

contestType: "Tough",

},

stoneedge: {

num: 444,

accuracy: 80,

basePower: 100,

category: "Physical",

desc: "Has a higher chance for a critical hit.",

shortDesc: "High critical hit ratio.",

name: "Stone Edge",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

critRatio: 2,

secondary: null,

target: "normal",

type: "Rock",

contestType: "Tough",

},

storedpower: {

num: 500,

accuracy: 100,

basePower: 20,

basePowerCallback(pokemon, target, move) {

return move.basePower + 20 \* pokemon.positiveBoosts();

},

category: "Special",

desc: "Power is equal to 20+(X\*20), where X is the user's total stat stage changes that are greater than 0.",

shortDesc: " + 20 power for each of the user's stat boosts.",

name: "Stored Power",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Clever",

},

stormthrow: {

num: 480,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "This move is always a critical hit unless the target is under the effect of Lucky Chant or has the Battle Armor or Shell Armor Abilities.",

shortDesc: "Always results in a critical hit.",

name: "Storm Throw",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

willCrit: true,

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

strangesteam: {

num: 790,

accuracy: 95,

basePower: 90,

category: "Special",

desc: "Has a 20% chance to confuse the target.",

shortDesc: "20% chance to confuse the target.",

name: "Strange Steam",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'confusion',

},

target: "normal",

type: "Fairy",

},

strength: {

num: 70,

accuracy: 100,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "Strength",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

strengthsap: {

num: 668,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack by 1 stage. The user restores its HP equal to the target's Attack stat calculated with its stat stage before this move was used. If Big Root is held by the user, the HP recovered is 1.3x normal, rounded half down. Fails if the target's Attack stat stage is -6.",

shortDesc: "User heals HP=target's Atk stat. Lowers Atk by 1.",

name: "Strength Sap",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, heal: 1},

onHit(target, source) {

if (target.boosts.atk === -6) return false;

const atk = target.getStat('atk', false, true);

const success = this.boost({atk: -1}, target, source, null, false, true);

return !!(this.heal(atk, source, target) || success);

},

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

stringshot: {

num: 81,

accuracy: 95,

basePower: 0,

category: "Status",

desc: "Lowers the target's Speed by 2 stages.",

shortDesc: "Lowers the foe(s) Speed by 2.",

name: "String Shot",

pp: 40,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

spe: -2,

},

secondary: null,

target: "allAdjacentFoes",

type: "Bug",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

struggle: {

num: 165,

accuracy: true,

basePower: 50,

category: "Physical",

desc: "Deals typeless damage to a random opposing Pokemon. If this move was successful, the user loses 1/4 of its maximum HP, rounded half up, and the Rock Head Ability does not prevent this. This move is automatically used if none of the user's known moves can be selected.",

shortDesc: "User loses 1/4 of its max HP.",

name: "Struggle",

pp: 1,

noPPBoosts: true,

priority: 0,

flags: {contact: 1, protect: 1},

noSketch: true,

onModifyMove(move, pokemon, target) {

move.type = '???';

this.add('-activate', pokemon, 'move: Struggle');

},

struggleRecoil: true,

secondary: null,

target: "randomNormal",

type: "Normal",

contestType: "Tough",

},

strugglebug: {

num: 522,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "Has a 100% chance to lower the target's Special Attack by 1 stage.",

shortDesc: "100% chance to lower the foe(s) Sp. Atk by 1.",

name: "Struggle Bug",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

spa: -1,

},

},

target: "allAdjacentFoes",

type: "Bug",

contestType: "Cute",

},

stuffcheeks: {

num: 747,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user eats its Berry and raises its Defense by 2 stages. This effect is not prevented by the Klutz or Unnerve Abilities, or the effects of Embargo or Magic Room. Fails if the user is not holding a Berry.",

shortDesc: "User eats its Berry and raises its Defense by 2.",

name: "Stuff Cheeks",

pp: 10,

priority: 0,

flags: {snatch: 1},

onTryHit(target, source, move) {

const item = source.getItem();

if (item.isBerry && source.eatItem(true)) {

this.boost({def: 2}, source, null, null, false, true);

} else {

return false;

}

},

secondary: null,

target: "self",

type: "Normal",

},

stunspore: {

num: 78,

accuracy: 75,

basePower: 0,

category: "Status",

desc: "Paralyzes the target.",

shortDesc: "Paralyzes the target.",

name: "Stun Spore",

pp: 30,

priority: 0,

flags: {powder: 1, protect: 1, reflectable: 1, mirror: 1},

status: 'par',

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

submission: {

num: 66,

accuracy: 80,

basePower: 80,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 1/4 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/4 recoil.",

name: "Submission",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [1, 4],

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

substitute: {

num: 164,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user takes 1/4 of its maximum HP, rounded down, and puts it into a substitute to take its place in battle. The substitute is removed once enough damage is inflicted on it, or if the user switches out or faints. Baton Pass can be used to transfer the substitute to an ally, and the substitute will keep its remaining HP. Until the substitute is broken, it receives damage from all attacks made by other Pokemon and shields the user from status effects and stat stage changes caused by other Pokemon. Sound-based moves and Pokemon with the Infiltrator Ability ignore substitutes. The user still takes normal damage from weather and status effects while behind its substitute. If the substitute breaks during a multi-hit attack, the user will take damage from any remaining hits. If a substitute is created while the user is trapped by a binding move, the binding effect ends immediately. Fails if the user does not have enough HP remaining to create a substitute without fainting, or if it already has a substitute.",

shortDesc: "User takes 1/4 its max HP to put in a substitute.",

name: "Substitute",

pp: 10,

priority: 0,

flags: {snatch: 1, nonsky: 1},

volatileStatus: 'substitute',

onTryHit(target) {

if (target.volatiles['substitute']) {

this.add('-fail', target, 'move: Substitute');

return null;

}

if (target.hp <= target.maxhp / 4 || target.maxhp === 1) { // Shedinja clause

this.add('-fail', target, 'move: Substitute', '[weak]');

return null;

}

},

onHit(target) {

this.directDamage(target.maxhp / 4);

},

condition: {

onStart(target) {

this.add('-start', target, 'Substitute');

this.effectData.hp = Math.floor(target.maxhp / 4);

if (target.volatiles['partiallytrapped']) {

this.add('-end', target, target.volatiles['partiallytrapped'].sourceEffect, '[partiallytrapped]', '[silent]');

delete target.volatiles['partiallytrapped'];

}

},

onTryPrimaryHitPriority: -1,

onTryPrimaryHit(target, source, move) {

if (target === source || move.flags['authentic'] || move.infiltrates) {

return;

}

let damage = this.getDamage(source, target, move);

if (!damage && damage !== 0) {

this.add('-fail', source);

this.attrLastMove('[still]');

return null;

}

damage = this.runEvent('SubDamage', target, source, move, damage);

if (!damage) {

return damage;

}

if (damage > target.volatiles['substitute'].hp) {

damage = target.volatiles['substitute'].hp as number;

}

target.volatiles['substitute'].hp -= damage;

source.lastDamage = damage;

if (target.volatiles['substitute'].hp <= 0) {

target.removeVolatile('substitute');

} else {

this.add('-activate', target, 'move: Substitute', '[damage]');

}

if (move.recoil) {

this.damage(this.calcRecoilDamage(damage, move), source, target, 'recoil');

}

if (move.drain) {

this.heal(Math.ceil(damage \* move.drain[0] / move.drain[1]), source, target, 'drain');

}

this.singleEvent('AfterSubDamage', move, null, target, source, move, damage);

this.runEvent('AfterSubDamage', target, source, move, damage);

return this.HIT\_SUBSTITUTE;

},

onEnd(target) {

this.add('-end', target, 'Substitute');

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

subzeroslammer: {

num: 650,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Subzero Slammer",

pp: 1,

priority: 0,

flags: {},

isZ: "iciumz",

secondary: null,

target: "normal",

type: "Ice",

contestType: "Cool",

},

suckerpunch: {

num: 389,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Fails if the target did not select a physical attack, special attack, or Me First for use this turn, or if the target moves before the user.",

shortDesc: "Usually goes first. Fails if target is not attacking.",

name: "Sucker Punch",

pp: 5,

priority: 1,

flags: {contact: 1, protect: 1, mirror: 1},

onTry(source, target) {

const action = this.queue.willMove(target);

const move = action?.choice === 'move' ? action.move : null;

if (!move || (move.category === 'Status' && move.id !== 'mefirst') || target.volatiles.mustrecharge) {

this.add('-fail', source);

this.attrLastMove('[still]');

return null;

}

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Clever",

},

sunnyday: {

num: 241,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the weather becomes Sunny Day. The damage of Fire-type attacks is multiplied by 1.5 and the damage of Water-type attacks is multiplied by 0.5 during the effect. Lasts for 8 turns if the user is holding Heat Rock. Fails if the current weather is Sunny Day.",

shortDesc: "For 5 turns, intense sunlight powers Fire moves.",

name: "Sunny Day",

pp: 5,

priority: 0,

flags: {},

weather: 'sunnyday',

secondary: null,

target: "all",

type: "Fire",

zMove: {boost: {spe: 1}},

contestType: "Beautiful",

},

sunsteelstrike: {

num: 713,

accuracy: 100,

basePower: 100,

category: "Physical",

desc: "This move and its effects ignore the Abilities of other Pokemon.",

shortDesc: "Ignores the Abilities of other Pokemon.",

name: "Sunsteel Strike",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

ignoreAbility: true,

secondary: null,

target: "normal",

type: "Steel",

contestType: "Cool",

},

superfang: {

num: 162,

accuracy: 90,

basePower: 0,

damageCallback(pokemon, target) {

return this.clampIntRange(target.getUndynamaxedHP() / 2, 1);

},

category: "Physical",

desc: "Deals damage to the target equal to half of its current HP, rounded down, but not less than 1 HP.",

shortDesc: "Does damage equal to 1/2 target's current HP.",

name: "Super Fang",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

superpower: {

num: 276,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "Lowers the user's Attack and Defense by 1 stage.",

shortDesc: "Lowers the user's Attack and Defense by 1.",

name: "Superpower",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

boosts: {

atk: -1,

def: -1,

},

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

supersonic: {

num: 48,

accuracy: 55,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to become confused.",

name: "Supersonic",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, sound: 1, authentic: 1},

volatileStatus: 'confusion',

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

supersonicskystrike: {

num: 626,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Supersonic Skystrike",

pp: 1,

priority: 0,

flags: {},

isZ: "flyiniumz",

secondary: null,

target: "normal",

type: "Flying",

contestType: "Cool",

},

surf: {

num: 57,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Damage doubles if the target is using Dive.",

shortDesc: "Hits adjacent Pokemon. Double damage on Dive.",

name: "Surf",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

secondary: null,

target: "allAdjacent",

type: "Water",

contestType: "Beautiful",

},

surgingstrikes: {

num: 818,

accuracy: 100,

basePower: 25,

category: "Physical",

desc: "Hits three times. This move is always a critical hit unless the target is under the effect of Lucky Chant or has the Battle Armor or Shell Armor Abilities.",

shortDesc: "Always results in a critical hit. Hits 3 times.",

name: "Surging Strikes",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1},

willCrit: true,

multihit: 3,

secondary: null,

target: "normal",

type: "Water",

zMove: {basePower: 140},

maxMove: {basePower: 130},

},

swagger: {

num: 207,

accuracy: 85,

basePower: 0,

category: "Status",

desc: "Raises the target's Attack by 2 stages and confuses it.",

shortDesc: "Raises the target's Attack by 2 and confuses it.",

name: "Swagger",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

volatileStatus: 'confusion',

boosts: {

atk: 2,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Cute",

},

swallow: {

num: 256,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores its HP based on its Stockpile count. Restores 1/4 of its maximum HP if it's 1, 1/2 of its maximum HP if it's 2, both rounded half down, and all of its HP if it's 3. Fails if the user's Stockpile count is 0. The user's Defense and Special Defense decrease by as many stages as Stockpile had increased them, and the user's Stockpile count resets to 0.",

shortDesc: "Heals the user based on uses of Stockpile.",

name: "Swallow",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

onTryHit(pokemon) {

if (!pokemon.volatiles['stockpile'] || !pokemon.volatiles['stockpile'].layers) return false;

},

onHit(pokemon) {

const healAmount = [0.25, 0.5, 1];

const healedBy = this.heal(this.modify(pokemon.maxhp, healAmount[(pokemon.volatiles['stockpile'].layers - 1)]));

pokemon.removeVolatile('stockpile');

return !!healedBy;

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Tough",

},

sweetkiss: {

num: 186,

accuracy: 75,

basePower: 0,

category: "Status",

shortDesc: "Causes the target to become confused.",

name: "Sweet Kiss",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'confusion',

secondary: null,

target: "normal",

type: "Fairy",

zMove: {boost: {spa: 1}},

contestType: "Cute",

},

sweetscent: {

num: 230,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's evasiveness by 2 stages.",

shortDesc: "Lowers the foe(s) evasiveness by 2.",

name: "Sweet Scent",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

evasion: -2,

},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

zMove: {boost: {accuracy: 1}},

contestType: "Cute",

},

swift: {

num: 129,

accuracy: true,

basePower: 60,

category: "Special",

desc: "This move does not check accuracy.",

shortDesc: "This move does not check accuracy. Hits foes.",

name: "Swift",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

contestType: "Cool",

},

switcheroo: {

num: 415,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "The user swaps its held item with the target's held item. Fails if either the user or the target is holding a Mail or Z-Crystal, if neither is holding an item, if the user is trying to give or take a Mega Stone to or from the species that can Mega Evolve with it, or if the user is trying to give or take a Blue Orb, a Red Orb, a Griseous Orb, a Plate, a Drive, or a Memory to or from a Kyogre, a Groudon, a Giratina, an Arceus, a Genesect, or a Silvally, respectively. The target is immune to this move if it has the Sticky Hold Ability.",

shortDesc: "User switches its held item with the target's.",

name: "Switcheroo",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

onTryImmunity(target) {

return !target.hasAbility('stickyhold');

},

onHit(target, source, move) {

const yourItem = target.takeItem(source);

const myItem = source.takeItem();

if (target.item || source.item || (!yourItem && !myItem)) {

if (yourItem) target.item = yourItem.id;

if (myItem) source.item = myItem.id;

return false;

}

if (

(myItem && !this.singleEvent('TakeItem', myItem, source.itemData, target, source, move, myItem)) ||

(yourItem && !this.singleEvent('TakeItem', yourItem, target.itemData, source, target, move, yourItem))

) {

if (yourItem) target.item = yourItem.id;

if (myItem) source.item = myItem.id;

return false;

}

this.add('-activate', source, 'move: Trick', '[of] ' + target);

if (myItem) {

target.setItem(myItem);

this.add('-item', target, myItem, '[from] move: Switcheroo');

} else {

this.add('-enditem', target, yourItem, '[silent]', '[from] move: Switcheroo');

}

if (yourItem) {

source.setItem(yourItem);

this.add('-item', source, yourItem, '[from] move: Switcheroo');

} else {

this.add('-enditem', source, myItem, '[silent]', '[from] move: Switcheroo');

}

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

swordsdance: {

num: 14,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack by 2 stages.",

shortDesc: "Raises the user's Attack by 2.",

name: "Swords Dance",

pp: 20,

priority: 0,

flags: {snatch: 1, dance: 1},

boosts: {

atk: 2,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

synchronoise: {

num: 485,

accuracy: 100,

basePower: 120,

category: "Special",

desc: "The target is immune if it does not share a type with the user.",

shortDesc: "Hits adjacent Pokemon sharing the user's type.",

isNonstandard: "Past",

name: "Synchronoise",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onTryImmunity(target, source) {

return target.hasType(source.getTypes());

},

secondary: null,

target: "allAdjacent",

type: "Psychic",

contestType: "Clever",

},

synthesis: {

num: 235,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user restores 1/2 of its maximum HP if Delta Stream or no weather conditions are in effect or if the user is holding Utility Umbrella, 2/3 of its maximum HP if the weather is Desolate Land or Sunny Day, and 1/4 of its maximum HP if the weather is Hail, Primordial Sea, Rain Dance, or Sandstorm, all rounded half down.",

shortDesc: "Heals the user by a weather-dependent amount.",

name: "Synthesis",

pp: 5,

priority: 0,

flags: {snatch: 1, heal: 1},

onHit(pokemon) {

let factor = 0.5;

switch (pokemon.effectiveWeather()) {

case 'sunnyday':

case 'desolateland':

factor = 0.667;

break;

case 'raindance':

case 'primordialsea':

case 'sandstorm':

case 'hail':

factor = 0.25;

break;

}

return !!this.heal(this.modify(pokemon.maxhp, factor));

},

secondary: null,

target: "self",

type: "Grass",

zMove: {effect: 'clearnegativeboost'},

contestType: "Clever",

},

tackle: {

num: 33,

accuracy: 100,

basePower: 40,

category: "Physical",

shortDesc: "No additional effect.",

name: "Tackle",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

tailglow: {

num: 294,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Special Attack by 3 stages.",

shortDesc: "Raises the user's Sp. Atk by 3.",

isNonstandard: "Past",

name: "Tail Glow",

pp: 20,

priority: 0,

flags: {snatch: 1},

boosts: {

spa: 3,

},

secondary: null,

target: "self",

type: "Bug",

zMove: {effect: 'clearnegativeboost'},

contestType: "Beautiful",

},

tailslap: {

num: 541,

accuracy: 85,

basePower: 25,

category: "Physical",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times.",

shortDesc: "Hits 2-5 times in one turn.",

name: "Tail Slap",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 140},

maxMove: {basePower: 130},

contestType: "Cute",

},

tailwhip: {

num: 39,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Defense by 1 stage.",

shortDesc: "Lowers the foe(s) Defense by 1.",

name: "Tail Whip",

pp: 30,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

boosts: {

def: -1,

},

secondary: null,

target: "allAdjacentFoes",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Cute",

},

tailwind: {

num: 366,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 4 turns, the user and its party members have their Speed doubled. Fails if this move is already in effect for the user's side.",

shortDesc: "For 4 turns, allies' Speed is doubled.",

name: "Tailwind",

pp: 15,

priority: 0,

flags: {snatch: 1},

sideCondition: 'tailwind',

condition: {

duration: 4,

durationCallback(target, source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 6;

}

return 4;

},

onStart(side) {

this.add('-sidestart', side, 'move: Tailwind');

},

onModifySpe(spe, pokemon) {

return this.chainModify(2);

},

onResidualOrder: 21,

onResidualSubOrder: 4,

onEnd(side) {

this.add('-sideend', side, 'move: Tailwind');

},

},

secondary: null,

target: "allySide",

type: "Flying",

zMove: {effect: 'crit2'},

contestType: "Cool",

},

takedown: {

num: 36,

accuracy: 85,

basePower: 90,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 1/4 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/4 recoil.",

name: "Take Down",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [1, 4],

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

tarshot: {

num: 749,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Speed by 1 stage. Until the target switches out, the effectiveness of Fire-type moves is doubled against it.",

shortDesc: "Target gets -1 Spe and becomes weaker to Fire.",

name: "Tar Shot",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'tarshot',

condition: {

onStart(pokemon) {

this.add('-start', pokemon, 'Tar Shot');

},

onEffectivenessPriority: -2,

onEffectiveness(typeMod, target, type, move) {

if (move.type !== 'Fire') return;

if (!target) return;

if (type !== target.getTypes()[0]) return;

return typeMod + 1;

},

},

boosts: {

spe: -1,

},

secondary: null,

target: "normal",

type: "Rock",

},

taunt: {

num: 269,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Prevents the target from using non-damaging moves for its next three turns. Pokemon with the Oblivious Ability or protected by the Aroma Veil Ability are immune.",

shortDesc: "Target can't use status moves its next 3 turns.",

name: "Taunt",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'taunt',

condition: {

duration: 3,

onStart(target) {

if (target.activeTurns && !this.queue.willMove(target)) {

this.effectData.duration++;

}

this.add('-start', target, 'move: Taunt');

},

onResidualOrder: 12,

onEnd(target) {

this.add('-end', target, 'move: Taunt');

},

onDisableMove(pokemon) {

for (const moveSlot of pokemon.moveSlots) {

const move = this.dex.getMove(moveSlot.id);

if (move.category === 'Status' && move.id !== 'mefirst') {

pokemon.disableMove(moveSlot.id);

}

}

},

onBeforeMovePriority: 5,

onBeforeMove(attacker, defender, move) {

if (!move.isZ && !move.isMax && move.category === 'Status' && move.id !== 'mefirst') {

this.add('cant', attacker, 'move: Taunt', move);

return false;

}

},

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {atk: 1}},

contestType: "Clever",

},

tearfullook: {

num: 715,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack and Special Attack by 1 stage.",

shortDesc: "Lowers the target's Attack and Sp. Atk by 1.",

name: "Tearful Look",

pp: 20,

priority: 0,

flags: {reflectable: 1, mirror: 1},

boosts: {

atk: -1,

spa: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

teatime: {

num: 752,

accuracy: true,

basePower: 0,

category: "Status",

desc: "All active Pokemon consume their held Berries. This effect is not prevented by substitutes, the Klutz or Unnerve Abilities, or the effects of Embargo or Magic Room. Fails if no active Pokemon is holding a Berry.",

shortDesc: "All active Pokemon consume held Berries.",

name: "Teatime",

pp: 10,

priority: 0,

flags: {authentic: 1},

onHitField(target, source, move) {

let result = false;

for (const active of this.getAllActive()) {

if (this.runEvent('Invulnerability', active, source, move) === false) {

this.add('-miss', source, active);

result = true;

} else {

const item = active.getItem();

if (active.hp && item.isBerry) {

// bypasses Unnerve

active.eatItem(true);

result = true;

}

}

}

return result;

},

secondary: null,

target: "all",

type: "Normal",

},

technoblast: {

num: 546,

accuracy: 100,

basePower: 120,

category: "Special",

desc: "This move's type depends on the user's held Drive.",

shortDesc: "Type varies based on the held Drive.",

isNonstandard: "Past",

name: "Techno Blast",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyType(move, pokemon) {

if (pokemon.ignoringItem()) return;

move.type = this.runEvent('Drive', pokemon, null, move, 'Normal');

},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cool",

},

tectonicrage: {

num: 630,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Tectonic Rage",

pp: 1,

priority: 0,

flags: {},

isZ: "groundiumz",

secondary: null,

target: "normal",

type: "Ground",

contestType: "Cool",

},

teeterdance: {

num: 298,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target to become confused.",

shortDesc: "Confuses adjacent Pokemon.",

name: "Teeter Dance",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1, dance: 1},

volatileStatus: 'confusion',

secondary: null,

target: "allAdjacent",

type: "Normal",

zMove: {boost: {spa: 1}},

contestType: "Cute",

},

telekinesis: {

num: 477,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 3 turns, the target cannot avoid any attacks made against it, other than OHKO moves, as long as it remains active. During the effect, the target is immune to Ground-type attacks and the effects of Spikes, Toxic Spikes, Sticky Web, and the Arena Trap Ability as long as it remains active. If the target uses Baton Pass, the replacement will gain the effect. Ingrain, Smack Down, Thousand Arrows, and Iron Ball override this move if the target is under any of their effects. Fails if the target is already under this effect or the effects of Ingrain, Smack Down, or Thousand Arrows. The target is immune to this move on use if its species is Diglett, Dugtrio, Alolan Diglett, Alolan Dugtrio, Sandygast, Palossand, or Gengar while Mega-Evolved. Mega Gengar cannot be under this effect by any means.",

shortDesc: "For 3 turns, target floats but moves can't miss it.",

isNonstandard: "Past",

name: "Telekinesis",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, gravity: 1, mystery: 1},

volatileStatus: 'telekinesis',

condition: {

duration: 3,

onStart(target) {

if (['Diglett', 'Dugtrio', 'Palossand', 'Sandygast'].includes(target.baseSpecies.baseSpecies) ||

target.baseSpecies.name === 'Gengar-Mega') {

this.add('-immune', target);

return null;

}

if (target.volatiles['smackdown'] || target.volatiles['ingrain']) return false;

this.add('-start', target, 'Telekinesis');

},

onAccuracyPriority: -1,

onAccuracy(accuracy, target, source, move) {

if (move && !move.ohko) return true;

},

onImmunity(type) {

if (type === 'Ground') return false;

},

onUpdate(pokemon) {

if (pokemon.baseSpecies.name === 'Gengar-Mega') {

delete pokemon.volatiles['telekinesis'];

this.add('-end', pokemon, 'Telekinesis', '[silent]');

}

},

onResidualOrder: 16,

onEnd(target) {

this.add('-end', target, 'Telekinesis');

},

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spa: 1}},

contestType: "Clever",

},

teleport: {

num: 100,

accuracy: true,

basePower: 0,

category: "Status",

desc: "If this move is successful and the user has not fainted, the user switches out even if it is trapped and is replaced immediately by a selected party member. The user does not switch out if there are no unfainted party members.",

shortDesc: "User switches out.",

name: "Teleport",

pp: 20,

priority: -6,

flags: {},

selfSwitch: true,

onTryHit: true,

secondary: null,

target: "self",

type: "Psychic",

zMove: {effect: 'heal'},

contestType: "Cool",

},

terrainpulse: {

num: 805,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "Power doubles if the user is grounded and a terrain is active, and this move's type changes to match. Electric type during Electric Terrain, Grass type during Grassy Terrain, Fairy type during Misty Terrain, and Psychic type during Psychic Terrain.",

shortDesc: "User on terrain: power doubles, type varies.",

name: "Terrain Pulse",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, pulse: 1},

onModifyType(move, pokemon) {

if (!pokemon.isGrounded()) return;

switch (this.field.terrain) {

case 'electricterrain':

move.type = 'Electric';

break;

case 'grassyterrain':

move.type = 'Grass';

break;

case 'mistyterrain':

move.type = 'Fairy';

break;

case 'psychicterrain':

move.type = 'Psychic';

break;

}

},

onModifyMove(move, pokemon) {

if (this.field.terrain && pokemon.isGrounded()) {

move.basePower \*= 2;

}

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

},

thief: {

num: 168,

accuracy: 100,

basePower: 60,

category: "Physical",

desc: "If this attack was successful and the user has not fainted, it steals the target's held item if the user is not holding one. The target's item is not stolen if it is a Mail or Z-Crystal, or if the target is a Kyogre holding a Blue Orb, a Groudon holding a Red Orb, a Giratina holding a Griseous Orb, an Arceus holding a Plate, a Genesect holding a Drive, a Silvally holding a Memory, or a Pokemon that can Mega Evolve holding the Mega Stone for its species. Items lost to this move cannot be regained with Recycle or the Harvest Ability.",

shortDesc: "If the user has no item, it steals the target's.",

name: "Thief",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onAfterHit(target, source, move) {

if (source.item || source.volatiles['gem']) {

return;

}

const yourItem = target.takeItem(source);

if (!yourItem) {

return;

}

if (!this.singleEvent('TakeItem', yourItem, target.itemData, source, target, move, yourItem) ||

!source.setItem(yourItem)) {

target.item = yourItem.id; // bypass setItem so we don't break choicelock or anything

return;

}

this.add('-enditem', target, yourItem, '[silent]', '[from] move: Thief', '[of] ' + source);

this.add('-item', source, yourItem, '[from] move: Thief', '[of] ' + target);

},

secondary: null,

target: "normal",

type: "Dark",

contestType: "Tough",

},

thousandarrows: {

num: 614,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "This move can hit airborne Pokemon, which includes Flying-type Pokemon, Pokemon with the Levitate Ability, Pokemon holding an Air Balloon, and Pokemon under the effect of Magnet Rise or Telekinesis. If the target is a Flying type and is not already grounded, this move deals neutral damage regardless of its other type(s). This move can hit a target using Bounce, Fly, or Sky Drop. If this move hits a target under the effect of Bounce, Fly, Magnet Rise, or Telekinesis, the effect ends. If the target is a Flying type that has not used Roost this turn or a Pokemon with the Levitate Ability, it loses its immunity to Ground-type attacks and the Arena Trap Ability as long as it remains active. During the effect, Magnet Rise fails for the target and Telekinesis fails against the target.",

shortDesc: "Grounds adjacent foes. First hit neutral on Flying.",

isNonstandard: "Past",

name: "Thousand Arrows",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onEffectiveness(typeMod, target, type, move) {

if (move.type !== 'Ground') return;

if (!target) return; // avoid crashing when called from a chat plugin

// ignore effectiveness if the target is Flying type and immune to Ground

if (!target.runImmunity('Ground')) {

if (target.hasType('Flying')) return 0;

}

},

volatileStatus: 'smackdown',

ignoreImmunity: {'Ground': true},

secondary: null,

target: "allAdjacentFoes",

type: "Ground",

zMove: {basePower: 180},

contestType: "Beautiful",

},

thousandwaves: {

num: 615,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "Prevents the target from switching out. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. If the target leaves the field using Baton Pass, the replacement will remain trapped. The effect ends if the user leaves the field.",

shortDesc: "Hits adjacent foes. Prevents them from switching.",

isNonstandard: "Past",

name: "Thousand Waves",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onHit(target, source, move) {

if (source.isActive) target.addVolatile('trapped', source, move, 'trapper');

},

secondary: null,

target: "allAdjacentFoes",

type: "Ground",

contestType: "Tough",

},

thrash: {

num: 37,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "The user spends two or three turns locked into this move and becomes confused immediately after its move on the last turn of the effect if it is not already. This move targets an opposing Pokemon at random on each turn. If the user is prevented from moving, is asleep at the beginning of a turn, or the attack is not successful against the target on the first turn of the effect or the second turn of a three-turn effect, the effect ends without causing confusion. If this move is called by Sleep Talk and the user is asleep, the move is used for one turn and does not confuse the user.",

shortDesc: "Lasts 2-3 turns. Confuses the user afterwards.",

name: "Thrash",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

volatileStatus: 'lockedmove',

},

onAfterMove(pokemon) {

if (pokemon.volatiles['lockedmove'] && pokemon.volatiles['lockedmove'].duration === 1) {

pokemon.removeVolatile('lockedmove');

}

},

secondary: null,

target: "randomNormal",

type: "Normal",

contestType: "Tough",

},

throatchop: {

num: 675,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "For 2 turns, the target cannot use sound-based moves.",

shortDesc: "For 2 turns, the target cannot use sound moves.",

name: "Throat Chop",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

condition: {

duration: 2,

onStart(target) {

this.add('-start', target, 'Throat Chop', '[silent]');

},

onDisableMove(pokemon) {

for (const moveSlot of pokemon.moveSlots) {

if (this.dex.getMove(moveSlot.id).flags['sound']) {

pokemon.disableMove(moveSlot.id);

}

}

},

onBeforeMovePriority: 6,

onBeforeMove(pokemon, target, move) {

if (!move.isZ && !move.isMax && move.flags['sound']) {

this.add('cant', pokemon, 'move: Throat Chop');

return false;

}

},

onResidualOrder: 22,

onEnd(target) {

this.add('-end', target, 'Throat Chop', '[silent]');

},

},

secondary: {

chance: 100,

onHit(target) {

target.addVolatile('throatchop');

},

},

target: "normal",

type: "Dark",

contestType: "Clever",

},

thunder: {

num: 87,

accuracy: 70,

basePower: 110,

category: "Special",

desc: "Has a 30% chance to paralyze the target. This move can hit a target using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop. If the weather is Primordial Sea or Rain Dance, this move does not check accuracy. If the weather is Desolate Land or Sunny Day, this move's accuracy is 50%. If this move is used against a Pokemon holding Utility Umbrella, this move's accuracy remains at 70%.",

shortDesc: "30% chance to paralyze. Can't miss in rain.",

name: "Thunder",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onModifyMove(move, pokemon, target) {

switch (target.effectiveWeather()) {

case 'raindance':

case 'primordialsea':

move.accuracy = true;

break;

case 'sunnyday':

case 'desolateland':

move.accuracy = 50;

break;

}

},

secondary: {

chance: 30,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

thunderbolt: {

num: 85,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "Has a 10% chance to paralyze the target.",

shortDesc: "10% chance to paralyze the target.",

name: "Thunderbolt",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

thunderfang: {

num: 422,

accuracy: 95,

basePower: 65,

category: "Physical",

desc: "Has a 10% chance to paralyze the target and a 10% chance to flinch it.",

shortDesc: "10% chance to paralyze. 10% chance to flinch.",

name: "Thunder Fang",

pp: 15,

priority: 0,

flags: {bite: 1, contact: 1, protect: 1, mirror: 1},

secondaries: [

{

chance: 10,

status: 'par',

}, {

chance: 10,

volatileStatus: 'flinch',

},

],

target: "normal",

type: "Electric",

contestType: "Cool",

},

thunderpunch: {

num: 9,

accuracy: 100,

basePower: 75,

category: "Physical",

desc: "Has a 10% chance to paralyze the target.",

shortDesc: "10% chance to paralyze the target.",

name: "Thunder Punch",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, punch: 1},

secondary: {

chance: 10,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

thundershock: {

num: 84,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 10% chance to paralyze the target.",

shortDesc: "10% chance to paralyze the target.",

name: "Thunder Shock",

pp: 30,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 10,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

thunderwave: {

num: 86,

accuracy: 90,

basePower: 0,

category: "Status",

desc: "Paralyzes the target. This move does not ignore type immunity.",

shortDesc: "Paralyzes the target.",

name: "Thunder Wave",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'par',

ignoreImmunity: false,

secondary: null,

target: "normal",

type: "Electric",

zMove: {boost: {spd: 1}},

contestType: "Cool",

},

tickle: {

num: 321,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack and Defense by 1 stage.",

shortDesc: "Lowers the target's Attack and Defense by 1.",

name: "Tickle",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

boosts: {

atk: -1,

def: -1,

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

topsyturvy: {

num: 576,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target's positive stat stages become negative and vice versa. Fails if all of the target's stat stages are 0.",

shortDesc: "Inverts the target's stat stages.",

name: "Topsy-Turvy",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target) {

let success = false;

let i: BoostName;

for (i in target.boosts) {

if (target.boosts[i] === 0) continue;

target.boosts[i] = -target.boosts[i];

success = true;

}

if (!success) return false;

this.add('-invertboost', target, '[from] move: Topsy-Turvy');

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {atk: 1}},

contestType: "Clever",

},

torment: {

num: 259,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Prevents the target from selecting the same move for use two turns in a row. This effect ends when the target is no longer active.",

shortDesc: "Target can't select the same move twice in a row.",

name: "Torment",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, authentic: 1},

volatileStatus: 'torment',

condition: {

noCopy: true,

onStart(pokemon) {

if (pokemon.volatiles['dynamax']) {

delete pokemon.volatiles['torment'];

return false;

}

this.add('-start', pokemon, 'Torment');

},

onEnd(pokemon) {

this.add('-end', pokemon, 'Torment');

},

onDisableMove(pokemon) {

if (pokemon.lastMove && pokemon.lastMove.id !== 'struggle') pokemon.disableMove(pokemon.lastMove.id);

},

},

secondary: null,

target: "normal",

type: "Dark",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

toxic: {

num: 92,

accuracy: 90,

basePower: 0,

category: "Status",

desc: "Badly poisons the target. If a Poison-type Pokemon uses this move, the target cannot avoid the attack, even if the target is in the middle of a two-turn move.",

shortDesc: "Badly poisons the target. Poison types can't miss.",

name: "Toxic",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

// No Guard-like effect for Poison-type users implemented in Scripts#tryMoveHit

status: 'tox',

secondary: null,

target: "normal",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

toxicspikes: {

num: 390,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Sets up a hazard on the opposing side of the field, poisoning each opposing Pokemon that switches in, unless it is a Flying-type Pokemon or has the Levitate Ability. Can be used up to two times before failing. Opposing Pokemon become poisoned with one layer and badly poisoned with two layers. Can be removed from the opposing side if any opposing Pokemon uses Rapid Spin or Defog successfully, is hit by Defog, or a grounded Poison-type Pokemon switches in. Safeguard prevents the opposing party from being poisoned on switch-in, but a substitute does not.",

shortDesc: "Poisons grounded foes on switch-in. Max 2 layers.",

name: "Toxic Spikes",

pp: 20,

priority: 0,

flags: {reflectable: 1, nonsky: 1},

sideCondition: 'toxicspikes',

condition: {

// this is a side condition

onStart(side) {

this.add('-sidestart', side, 'move: Toxic Spikes');

this.effectData.layers = 1;

},

onRestart(side) {

if (this.effectData.layers >= 2) return false;

this.add('-sidestart', side, 'move: Toxic Spikes');

this.effectData.layers++;

},

onSwitchIn(pokemon) {

if (!pokemon.isGrounded()) return;

if (pokemon.hasType('Poison')) {

this.add('-sideend', pokemon.side, 'move: Toxic Spikes', '[of] ' + pokemon);

pokemon.side.removeSideCondition('toxicspikes');

} else if (pokemon.hasType('Steel') || pokemon.hasItem('heavydutyboots')) {

return;

} else if (this.effectData.layers >= 2) {

pokemon.trySetStatus('tox', pokemon.side.foe.active[0]);

} else {

pokemon.trySetStatus('psn', pokemon.side.foe.active[0]);

}

},

},

secondary: null,

target: "foeSide",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

toxicthread: {

num: 672,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Speed by 1 stage and poisons it.",

shortDesc: "Lowers the target's Speed by 1 and poisons it.",

isNonstandard: "Past",

name: "Toxic Thread",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'psn',

boosts: {

spe: -1,

},

secondary: null,

target: "normal",

type: "Poison",

zMove: {boost: {spe: 1}},

contestType: "Tough",

},

transform: {

num: 144,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user transforms into the target. The target's current stats, stat stages, types, moves, Ability, weight, gender, and sprite are copied. The user's level and HP remain the same and each copied move receives only 5 PP, with a maximum of 5 PP each. The user can no longer change formes if it would have the ability to do so. This move fails if it hits a substitute, if either the user or the target is already transformed, or if either is behind an Illusion.",

shortDesc: "Copies target's stats, moves, types, and Ability.",

name: "Transform",

pp: 10,

priority: 0,

flags: {mystery: 1},

onHit(target, pokemon) {

if (!pokemon.transformInto(target)) {

return false;

}

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {effect: 'heal'},

contestType: "Clever",

},

triattack: {

num: 161,

accuracy: 100,

basePower: 80,

category: "Special",

desc: "Has a 20% chance to either burn, freeze, or paralyze the target.",

shortDesc: "20% chance to paralyze or burn or freeze target.",

name: "Tri Attack",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

onHit(target, source) {

const result = this.random(3);

if (result === 0) {

target.trySetStatus('brn', source);

} else if (result === 1) {

target.trySetStatus('par', source);

} else {

target.trySetStatus('frz', source);

}

},

},

target: "normal",

type: "Normal",

contestType: "Beautiful",

},

trick: {

num: 271,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "The user swaps its held item with the target's held item. Fails if either the user or the target is holding a Mail or Z-Crystal, if neither is holding an item, if the user is trying to give or take a Mega Stone to or from the species that can Mega Evolve with it, or if the user is trying to give or take a Blue Orb, a Red Orb, a Griseous Orb, a Plate, a Drive, or a Memory to or from a Kyogre, a Groudon, a Giratina, an Arceus, a Genesect, or a Silvally, respectively. The target is immune to this move if it has the Sticky Hold Ability.",

shortDesc: "User switches its held item with the target's.",

name: "Trick",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, mystery: 1},

onTryImmunity(target) {

return !target.hasAbility('stickyhold');

},

onHit(target, source, move) {

const yourItem = target.takeItem(source);

const myItem = source.takeItem();

if (target.item || source.item || (!yourItem && !myItem)) {

if (yourItem) target.item = yourItem.id;

if (myItem) source.item = myItem.id;

return false;

}

if (

(myItem && !this.singleEvent('TakeItem', myItem, source.itemData, target, source, move, myItem)) ||

(yourItem && !this.singleEvent('TakeItem', yourItem, target.itemData, source, target, move, yourItem))

) {

if (yourItem) target.item = yourItem.id;

if (myItem) source.item = myItem.id;

return false;

}

this.add('-activate', source, 'move: Trick', '[of] ' + target);

if (myItem) {

target.setItem(myItem);

this.add('-item', target, myItem, '[from] move: Trick');

} else {

this.add('-enditem', target, yourItem, '[silent]', '[from] move: Trick');

}

if (yourItem) {

source.setItem(yourItem);

this.add('-item', source, yourItem, '[from] move: Trick');

} else {

this.add('-enditem', source, myItem, '[silent]', '[from] move: Trick');

}

},

secondary: null,

target: "normal",

type: "Psychic",

zMove: {boost: {spe: 2}},

contestType: "Clever",

},

trickortreat: {

num: 567,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the Ghost type to be added to the target, effectively making it have two or three types. Fails if the target is already a Ghost type. If Forest's Curse adds a type to the target, it replaces the type added by this move and vice versa.",

shortDesc: "Adds Ghost to the target's type(s).",

name: "Trick-or-Treat",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onHit(target) {

if (target.hasType('Ghost')) return false;

if (!target.addType('Ghost')) return false;

this.add('-start', target, 'typeadd', 'Ghost', '[from] move: Trick-or-Treat');

if (target.side.active.length === 2 && target.position === 1) {

// Curse Glitch

const action = this.queue.willMove(target);

if (action && action.move.id === 'curse') {

action.targetLoc = -1;

}

}

},

secondary: null,

target: "normal",

type: "Ghost",

zMove: {boost: {atk: 1, def: 1, spa: 1, spd: 1, spe: 1}},

contestType: "Cute",

},

trickroom: {

num: 433,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, the Speed of every Pokemon is recalculated for the purposes of determining turn order. During the effect, each Pokemon's Speed is considered to be (10000 - its normal Speed), and if this value is greater than 8191, 8192 is subtracted from it. If this move is used during the effect, the effect ends.",

shortDesc: "Goes last. For 5 turns, turn order is reversed.",

name: "Trick Room",

pp: 5,

priority: -7,

flags: {mirror: 1},

pseudoWeather: 'trickroom',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onStart(target, source) {

this.add('-fieldstart', 'move: Trick Room', '[of] ' + source);

},

onRestart(target, source) {

this.field.removePseudoWeather('trickroom');

},

// Speed modification is changed in Pokemon.getActionSpeed() in sim/pokemon.js

onResidualOrder: 23,

onEnd() {

this.add('-fieldend', 'move: Trick Room');

},

},

secondary: null,

target: "all",

type: "Psychic",

zMove: {boost: {accuracy: 1}},

contestType: "Clever",

},

tripleaxel: {

num: 813,

accuracy: 90,

basePower: 20,

basePowerCallback(pokemon, target, move) {

return 20 \* move.hit;

},

category: "Physical",

desc: "Hits three times. Power increases to 40 for the second hit and 60 for the third. This move checks accuracy for each hit, and the attack ends if the target avoids a hit. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit three times.",

shortDesc: "Hits 3 times. Each hit can miss, but power rises.",

name: "Triple Axel",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 3,

multiaccuracy: true,

secondary: null,

target: "normal",

type: "Ice",

zMove: {basePower: 120},

maxMove: {basePower: 140},

},

triplekick: {

num: 167,

accuracy: 90,

basePower: 10,

basePowerCallback(pokemon, target, move) {

return 10 \* move.hit;

},

category: "Physical",

desc: "Hits three times. Power increases to 20 for the second hit and 30 for the third. This move checks accuracy for each hit, and the attack ends if the target avoids a hit. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit three times.",

shortDesc: "Hits 3 times. Each hit can miss, but power rises.",

name: "Triple Kick",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

multihit: 3,

multiaccuracy: true,

secondary: null,

target: "normal",

type: "Fighting",

zMove: {basePower: 120},

maxMove: {basePower: 80},

contestType: "Cool",

},

tropkick: {

num: 688,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "Has a 100% chance to lower the target's Attack by 1 stage.",

shortDesc: "100% chance to lower the target's Attack by 1.",

name: "Trop Kick",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

boosts: {

atk: -1,

},

},

target: "normal",

type: "Grass",

contestType: "Cute",

},

trumpcard: {

num: 376,

accuracy: true,

basePower: 0,

basePowerCallback(source, target, move) {

const callerMoveId = move.sourceEffect || move.id;

const moveSlot = callerMoveId === 'instruct' ? source.getMoveData(move.id) : source.getMoveData(callerMoveId);

if (!moveSlot) return 40;

switch (moveSlot.pp) {

case 0:

return 200;

case 1:

return 80;

case 2:

return 60;

case 3:

return 50;

default:

return 40;

}

},

category: "Special",

desc: "The power of this move is based on the amount of PP remaining after normal PP reduction and the Pressure Ability resolve. 200 power for 0 PP, 80 power for 1 PP, 60 power for 2 PP, 50 power for 3 PP, and 40 power for 4 or more PP.",

shortDesc: "More power the fewer PP this move has left.",

isNonstandard: "Past",

name: "Trump Card",

pp: 5,

noPPBoosts: true,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Cool",

},

twineedle: {

num: 41,

accuracy: 100,

basePower: 25,

category: "Physical",

desc: "Hits twice, with each hit having a 20% chance to poison the target. If the first hit breaks the target's substitute, it will take damage for the second hit.",

shortDesc: "Hits 2 times. Each hit has 20% chance to poison.",

isNonstandard: "Past",

name: "Twineedle",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

multihit: 2,

secondary: {

chance: 20,

status: 'psn',

},

target: "normal",

type: "Bug",

maxMove: {basePower: 100},

contestType: "Cool",

},

twinkletackle: {

num: 656,

accuracy: true,

basePower: 1,

category: "Physical",

shortDesc: "Power is equal to the base move's Z-Power.",

isNonstandard: "Past",

name: "Twinkle Tackle",

pp: 1,

priority: 0,

flags: {},

isZ: "fairiumz",

secondary: null,

target: "normal",

type: "Fairy",

contestType: "Cool",

},

twister: {

num: 239,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "Has a 20% chance to flinch the target. Power doubles if the target is using Bounce, Fly, or Sky Drop, or is under the effect of Sky Drop.",

shortDesc: "20% chance to flinch the foe(s).",

name: "Twister",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'flinch',

},

target: "allAdjacentFoes",

type: "Dragon",

contestType: "Cool",

},

uturn: {

num: 369,

accuracy: 100,

basePower: 70,

category: "Physical",

desc: "If this move is successful and the user has not fainted, the user switches out even if it is trapped and is replaced immediately by a selected party member. The user does not switch out if there are no unfainted party members, or if the target switched out using an Eject Button or through the effect of the Emergency Exit or Wimp Out Abilities.",

shortDesc: "User switches out after damaging the target.",

name: "U-turn",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

selfSwitch: true,

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cute",

},

uproar: {

num: 253,

accuracy: 100,

basePower: 90,

category: "Special",

desc: "The user spends three turns locked into this move. This move targets an opponent at random on each turn. On the first of the three turns, all sleeping active Pokemon wake up. During the three turns, no active Pokemon can fall asleep by any means, and Pokemon switched in during the effect do not wake up. If the user is prevented from moving or the attack is not successful against the target during one of the turns, the effect ends.",

shortDesc: "Lasts 3 turns. Active Pokemon cannot fall asleep.",

name: "Uproar",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, sound: 1, authentic: 1},

self: {

volatileStatus: 'uproar',

},

onTryHit(target) {

for (const [i, allyActive] of target.side.active.entries()) {

if (allyActive && allyActive.status === 'slp') allyActive.cureStatus();

const foeActive = target.side.foe.active[i];

if (foeActive && foeActive.status === 'slp') foeActive.cureStatus();

}

},

condition: {

duration: 3,

onStart(target) {

this.add('-start', target, 'Uproar');

},

onResidual(target) {

if (target.lastMove && target.lastMove.id === 'struggle') {

// don't lock

delete target.volatiles['uproar'];

}

this.add('-start', target, 'Uproar', '[upkeep]');

},

onEnd(target) {

this.add('-end', target, 'Uproar');

},

onLockMove: 'uproar',

onAnySetStatus(status, pokemon) {

if (status.id === 'slp') {

if (pokemon === this.effectData.target) {

this.add('-fail', pokemon, 'slp', '[from] Uproar', '[msg]');

} else {

this.add('-fail', pokemon, 'slp', '[from] Uproar');

}

return null;

}

},

},

secondary: null,

target: "randomNormal",

type: "Normal",

contestType: "Cute",

},

vacuumwave: {

num: 410,

accuracy: 100,

basePower: 40,

category: "Special",

desc: "No additional effect.",

shortDesc: "Usually goes first.",

name: "Vacuum Wave",

pp: 30,

priority: 1,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

vcreate: {

num: 557,

accuracy: 95,

basePower: 180,

category: "Physical",

desc: "Lowers the user's Speed, Defense, and Special Defense by 1 stage.",

shortDesc: "Lowers the user's Defense, Sp. Def, Speed by 1.",

name: "V-create",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

self: {

boosts: {

spe: -1,

def: -1,

spd: -1,

},

},

secondary: null,

target: "normal",

type: "Fire",

zMove: {basePower: 220},

contestType: "Cool",

},

veeveevolley: {

num: 741,

accuracy: true,

basePower: 0,

basePowerCallback(pokemon) {

return Math.floor((pokemon.happiness \* 10) / 25) || 1;

},

category: "Physical",

desc: "Power is equal to the greater of (user's Happiness \* 2/5), rounded down, or 1.",

shortDesc: "Max happiness: 102 power. Can't miss.",

isNonstandard: "LGPE",

name: "Veevee Volley",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Cute",

},

venomdrench: {

num: 599,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Lowers the target's Attack, Special Attack, and Speed by 1 stage if the target is poisoned. Fails if the target is not poisoned.",

shortDesc: "Lowers Atk/Sp. Atk/Speed of poisoned foes by 1.",

name: "Venom Drench",

pp: 20,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

onHit(target, source, move) {

if (target.status === 'psn' || target.status === 'tox') {

return !!this.boost({atk: -1, spa: -1, spe: -1}, target, source, move);

}

return false;

},

secondary: null,

target: "allAdjacentFoes",

type: "Poison",

zMove: {boost: {def: 1}},

contestType: "Clever",

},

venoshock: {

num: 474,

accuracy: 100,

basePower: 65,

category: "Special",

desc: "Power doubles if the target is poisoned.",

shortDesc: "Power doubles if the target is poisoned.",

name: "Venoshock",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1},

onBasePower(basePower, pokemon, target) {

if (target.status === 'psn' || target.status === 'tox') {

return this.chainModify(2);

}

},

secondary: null,

target: "normal",

type: "Poison",

contestType: "Beautiful",

},

vinewhip: {

num: 22,

accuracy: 100,

basePower: 45,

category: "Physical",

shortDesc: "No additional effect.",

name: "Vine Whip",

pp: 25,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Grass",

contestType: "Cool",

},

visegrip: {

num: 11,

accuracy: 100,

basePower: 55,

category: "Physical",

shortDesc: "No additional effect.",

name: "Vise Grip",

pp: 30,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

vitalthrow: {

num: 233,

accuracy: true,

basePower: 70,

category: "Physical",

desc: "This move does not check accuracy.",

shortDesc: "This move does not check accuracy. Goes last.",

name: "Vital Throw",

pp: 10,

priority: -1,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Cool",

},

voltswitch: {

num: 521,

accuracy: 100,

basePower: 70,

category: "Special",

desc: "If this move is successful and the user has not fainted, the user switches out even if it is trapped and is replaced immediately by a selected party member. The user does not switch out if there are no unfainted party members, or if the target switched out using an Eject Button or through the effect of the Emergency Exit or Wimp Out Abilities.",

shortDesc: "User switches out after damaging the target.",

name: "Volt Switch",

pp: 20,

priority: 0,

flags: {protect: 1, mirror: 1},

selfSwitch: true,

secondary: null,

target: "normal",

type: "Electric",

contestType: "Cool",

},

volttackle: {

num: 344,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "Has a 10% chance to paralyze the target. If the target lost HP, the user takes recoil damage equal to 33% the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 33% recoil. 10% chance to paralyze target.",

name: "Volt Tackle",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [33, 100],

secondary: {

chance: 10,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

wakeupslap: {

num: 358,

accuracy: 100,

basePower: 70,

basePowerCallback(pokemon, target, move) {

if (target.status === 'slp' || target.hasAbility('comatose')) return move.basePower \* 2;

return move.basePower;

},

category: "Physical",

desc: "Power doubles if the target is asleep. If the user has not fainted, the target wakes up.",

shortDesc: "Power doubles if target is asleep, and wakes it.",

isNonstandard: "Past",

name: "Wake-Up Slap",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

onHit(target) {

if (target.status === 'slp') target.cureStatus();

},

secondary: null,

target: "normal",

type: "Fighting",

contestType: "Tough",

},

waterfall: {

num: 127,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 20% chance to flinch the target.",

shortDesc: "20% chance to flinch the target.",

name: "Waterfall",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'flinch',

},

target: "normal",

type: "Water",

contestType: "Tough",

},

watergun: {

num: 55,

accuracy: 100,

basePower: 40,

category: "Special",

shortDesc: "No additional effect.",

name: "Water Gun",

pp: 25,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Water",

contestType: "Cute",

},

waterpledge: {

num: 518,

accuracy: 100,

basePower: 80,

basePowerCallback(target, source, move) {

if (['firepledge', 'grasspledge'].includes(move.sourceEffect)) {

this.add('-combine');

return 150;

}

return 80;

},

category: "Special",

desc: "If one of the user's allies chose to use Fire Pledge or Grass Pledge this turn and has not moved yet, it takes its turn immediately after the user and the user's move does nothing. If combined with Fire Pledge, the ally uses Water Pledge with 150 power and a rainbow appears on the user's side for 4 turns, which doubles secondary effect chances but does not stack with the Serene Grace Ability. If combined with Grass Pledge, the ally uses Grass Pledge with 150 power and a swamp appears on the target's side for 4 turns, which quarters the Speed of each Pokemon on that side. When used as a combined move, this move gains STAB no matter what the user's type is. This move does not consume the user's Water Gem, and cannot be redirected by the Storm Drain Ability.",

shortDesc: "Use with Grass or Fire Pledge for added effect.",

name: "Water Pledge",

pp: 10,

priority: 0,

flags: {protect: 1, mirror: 1, nonsky: 1},

onPrepareHit(target, source, move) {

for (const action of this.queue) {

if (action.choice !== 'move') continue;

const otherMove = action.move;

const otherMoveUser = action.pokemon;

if (

!otherMove || !action.pokemon || !otherMoveUser.isActive ||

otherMoveUser.fainted || action.maxMove || action.zmove

) {

continue;

}

if (otherMoveUser.side === source.side && ['firepledge', 'grasspledge'].includes(otherMove.id)) {

this.queue.prioritizeAction(action, move);

this.add('-waiting', source, otherMoveUser);

return null;

}

}

},

onModifyMove(move) {

if (move.sourceEffect === 'grasspledge') {

move.type = 'Grass';

move.forceSTAB = true;

move.sideCondition = 'grasspledge';

}

if (move.sourceEffect === 'firepledge') {

move.type = 'Water';

move.forceSTAB = true;

move.self = {sideCondition: 'waterpledge'};

}

},

condition: {

duration: 4,

onStart(targetSide) {

this.add('-sidestart', targetSide, 'Water Pledge');

},

onEnd(targetSide) {

this.add('-sideend', targetSide, 'Water Pledge');

},

onModifyMove(move) {

if (move.secondaries && move.id !== 'secretpower') {

this.debug('doubling secondary chance');

for (const secondary of move.secondaries) {

if (secondary.chance) secondary.chance \*= 2;

}

}

},

},

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

waterpulse: {

num: 352,

accuracy: 100,

basePower: 60,

category: "Special",

desc: "Has a 20% chance to confuse the target.",

shortDesc: "20% chance to confuse the target.",

name: "Water Pulse",

pp: 20,

priority: 0,

flags: {protect: 1, pulse: 1, mirror: 1, distance: 1},

secondary: {

chance: 20,

volatileStatus: 'confusion',

},

target: "any",

type: "Water",

contestType: "Beautiful",

},

watershuriken: {

num: 594,

accuracy: 100,

basePower: 15,

basePowerCallback(pokemon, target, move) {

if (pokemon.species.name === 'Greninja-Ash' && pokemon.hasAbility('battlebond')) {

return move.basePower + 5;

}

return move.basePower;

},

category: "Special",

desc: "Hits two to five times. Has a 1/3 chance to hit two or three times, and a 1/6 chance to hit four or five times. If one of the hits breaks the target's substitute, it will take damage for the remaining hits. If the user has the Skill Link Ability, this move will always hit five times. If the user is an Ash-Greninja with the Battle Bond Ability, this move has a power of 20 and always hits three times.",

shortDesc: "Usually goes first. Hits 2-5 times in one turn.",

name: "Water Shuriken",

pp: 20,

priority: 1,

flags: {protect: 1, mirror: 1},

multihit: [2, 5],

secondary: null,

target: "normal",

type: "Water",

contestType: "Cool",

},

watersport: {

num: 346,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, all Fire-type attacks used by any active Pokemon have their power multiplied by 0.33. Fails if this effect is already active.",

shortDesc: "For 5 turns, Fire-type attacks have 1/3 power.",

isNonstandard: "Past",

name: "Water Sport",

pp: 15,

priority: 0,

flags: {nonsky: 1},

pseudoWeather: 'watersport',

condition: {

duration: 5,

onStart(side, source) {

this.add('-fieldstart', 'move: Water Sport', '[of] ' + source);

},

onBasePowerPriority: 1,

onBasePower(basePower, attacker, defender, move) {

if (move.type === 'Fire') {

this.debug('water sport weaken');

return this.chainModify([0x548, 0x1000]);

}

},

onResidualOrder: 21,

onEnd() {

this.add('-fieldend', 'move: Water Sport');

},

},

secondary: null,

target: "all",

type: "Water",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

waterspout: {

num: 323,

accuracy: 100,

basePower: 150,

basePowerCallback(pokemon, target, move) {

return move.basePower \* pokemon.hp / pokemon.maxhp;

},

category: "Special",

desc: "Power is equal to (user's current HP \* 150 / user's maximum HP), rounded down, but not less than 1.",

shortDesc: "Less power as user's HP decreases. Hits foe(s).",

name: "Water Spout",

pp: 5,

priority: 0,

flags: {protect: 1, mirror: 1},

secondary: null,

target: "allAdjacentFoes",

type: "Water",

contestType: "Beautiful",

},

weatherball: {

num: 311,

accuracy: 100,

basePower: 50,

category: "Special",

desc: "Power doubles if a weather condition other than Delta Stream is active, and this move's type changes to match. Ice type during Hail, Water type during Primordial Sea or Rain Dance, Rock type during Sandstorm, and Fire type during Desolate Land or Sunny Day. If the user is holding Utility Umbrella and uses Weather Ball during Primordial Sea, Rain Dance, Desolate Land, or Sunny Day, the move is still Normal-type and does not have a base power boost.",

shortDesc: "Power doubles and type varies in each weather.",

name: "Weather Ball",

pp: 10,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

onModifyType(move, pokemon) {

switch (pokemon.effectiveWeather()) {

case 'sunnyday':

case 'desolateland':

move.type = 'Fire';

break;

case 'raindance':

case 'primordialsea':

move.type = 'Water';

break;

case 'sandstorm':

move.type = 'Rock';

break;

case 'hail':

move.type = 'Ice';

break;

}

},

onModifyMove(move, pokemon) {

switch (pokemon.effectiveWeather()) {

case 'sunnyday':

case 'desolateland':

move.basePower \*= 2;

break;

case 'raindance':

case 'primordialsea':

move.basePower \*= 2;

break;

case 'sandstorm':

move.basePower \*= 2;

break;

case 'hail':

move.basePower \*= 2;

break;

}

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 160},

maxMove: {basePower: 130},

contestType: "Beautiful",

},

whirlpool: {

num: 250,

accuracy: 85,

basePower: 35,

category: "Special",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Whirlpool",

pp: 15,

priority: 0,

flags: {protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Water",

contestType: "Beautiful",

},

whirlwind: {

num: 18,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The target is forced to switch out and be replaced with a random unfainted ally. Fails if the target is the last unfainted Pokemon in its party, or if the target used Ingrain previously or has the Suction Cups Ability.",

shortDesc: "Forces the target to switch to a random ally.",

name: "Whirlwind",

pp: 20,

priority: -6,

flags: {reflectable: 1, mirror: 1, authentic: 1, mystery: 1},

forceSwitch: true,

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

wickedblow: {

num: 817,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "This move is always a critical hit unless the target is under the effect of Lucky Chant or has the Battle Armor or Shell Armor Abilities.",

shortDesc: "Always results in a critical hit.",

name: "Wicked Blow",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1},

willCrit: true,

secondary: null,

target: "normal",

type: "Dark",

},

wideguard: {

num: 469,

accuracy: true,

basePower: 0,

category: "Status",

desc: "The user and its party members are protected from moves made by other Pokemon, including allies, during this turn that target all adjacent foes or all adjacent Pokemon. This move modifies the same 1/X chance of being successful used by other protection moves, where X starts at 1 and triples each time this move is successfully used, but does not use the chance to check for failure. X resets to 1 if this move fails, if the user's last move used is not Baneful Bunker, Detect, Endure, King's Shield, Obstruct, Protect, Quick Guard, Spiky Shield, or Wide Guard, or if it was one of those moves and the user's protection was broken. Fails if the user moves last this turn or if this move is already in effect for the user's side.",

shortDesc: "Protects allies from multi-target moves this turn.",

name: "Wide Guard",

pp: 10,

priority: 3,

flags: {snatch: 1},

sideCondition: 'wideguard',

onTryHitSide(side, source) {

return !!this.queue.willAct();

},

onHitSide(side, source) {

source.addVolatile('stall');

},

condition: {

duration: 1,

onStart(target, source) {

this.add('-singleturn', source, 'Wide Guard');

},

onTryHitPriority: 4,

onTryHit(target, source, move) {

// Wide Guard blocks all spread moves

if (move?.target !== 'allAdjacent' && move.target !== 'allAdjacentFoes') {

return;

}

if (move.isZ || move.isMax) {

if (['gmaxoneblow', 'gmaxrapidflow'].includes(move.id)) return;

target.getMoveHitData(move).zBrokeProtect = true;

return;

}

this.add('-activate', target, 'move: Wide Guard');

const lockedmove = source.getVolatile('lockedmove');

if (lockedmove) {

// Outrage counter is reset

if (source.volatiles['lockedmove'].duration === 2) {

delete source.volatiles['lockedmove'];

}

}

return this.NOT\_FAIL;

},

},

secondary: null,

target: "allySide",

type: "Rock",

zMove: {boost: {def: 1}},

contestType: "Tough",

},

wildcharge: {

num: 528,

accuracy: 100,

basePower: 90,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 1/4 the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 1/4 recoil.",

name: "Wild Charge",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [1, 4],

secondary: null,

target: "normal",

type: "Electric",

contestType: "Tough",

},

willowisp: {

num: 261,

accuracy: 85,

basePower: 0,

category: "Status",

desc: "Burns the target.",

shortDesc: "Burns the target.",

name: "Will-O-Wisp",

pp: 15,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

status: 'brn',

secondary: null,

target: "normal",

type: "Fire",

zMove: {boost: {atk: 1}},

contestType: "Beautiful",

},

wingattack: {

num: 17,

accuracy: 100,

basePower: 60,

category: "Physical",

shortDesc: "No additional effect.",

name: "Wing Attack",

pp: 35,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1, distance: 1},

secondary: null,

target: "any",

type: "Flying",

contestType: "Cool",

},

wish: {

num: 273,

accuracy: true,

basePower: 0,

category: "Status",

desc: "At the end of the next turn, the Pokemon at the user's position has 1/2 of the user's maximum HP restored to it, rounded half up. Fails if this move is already in effect for the user's position.",

shortDesc: "Next turn, 50% of the user's max HP is restored.",

name: "Wish",

pp: 10,

priority: 0,

flags: {snatch: 1, heal: 1},

slotCondition: 'Wish',

condition: {

duration: 2,

onStart(pokemon, source) {

this.effectData.hp = source.maxhp / 2;

},

onResidualOrder: 4,

onEnd(target) {

if (target && !target.fainted) {

const damage = this.heal(this.effectData.hp, target, target);

if (damage) this.add('-heal', target, target.getHealth, '[from] move: Wish', '[wisher] ' + this.effectData.source.name);

}

},

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {spd: 1}},

contestType: "Cute",

},

withdraw: {

num: 110,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Defense by 1 stage.",

shortDesc: "Raises the user's Defense by 1.",

name: "Withdraw",

pp: 40,

priority: 0,

flags: {snatch: 1},

boosts: {

def: 1,

},

secondary: null,

target: "self",

type: "Water",

zMove: {boost: {def: 1}},

contestType: "Cute",

},

wonderroom: {

num: 472,

accuracy: true,

basePower: 0,

category: "Status",

desc: "For 5 turns, all active Pokemon have their Defense and Special Defense stats swapped. Stat stage changes are unaffected. If this move is used during the effect, the effect ends.",

shortDesc: "For 5 turns, all Defense and Sp. Def stats switch.",

name: "Wonder Room",

pp: 10,

priority: 0,

flags: {mirror: 1},

pseudoWeather: 'wonderroom',

condition: {

duration: 5,

durationCallback(source, effect) {

if (source?.hasAbility('persistent')) {

this.add('-activate', source, 'ability: Persistent', effect);

return 7;

}

return 5;

},

onStart(side, source) {

this.add('-fieldstart', 'move: Wonder Room', '[of] ' + source);

},

onRestart(target, source) {

this.field.removePseudoWeather('wonderroom');

},

// Swapping defenses implemented in sim/pokemon.js:Pokemon#calculateStat and Pokemon#getStat

onResidualOrder: 24,

onEnd() {

this.add('-fieldend', 'move: Wonder Room');

},

},

secondary: null,

target: "all",

type: "Psychic",

zMove: {boost: {spd: 1}},

contestType: "Clever",

},

woodhammer: {

num: 452,

accuracy: 100,

basePower: 120,

category: "Physical",

desc: "If the target lost HP, the user takes recoil damage equal to 33% the HP lost by the target, rounded half up, but not less than 1 HP.",

shortDesc: "Has 33% recoil.",

name: "Wood Hammer",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

recoil: [33, 100],

secondary: null,

target: "normal",

type: "Grass",

contestType: "Tough",

},

workup: {

num: 526,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Raises the user's Attack and Special Attack by 1 stage.",

shortDesc: "Raises the user's Attack and Sp. Atk by 1.",

name: "Work Up",

pp: 30,

priority: 0,

flags: {snatch: 1},

boosts: {

atk: 1,

spa: 1,

},

secondary: null,

target: "self",

type: "Normal",

zMove: {boost: {atk: 1}},

contestType: "Tough",

},

worryseed: {

num: 388,

accuracy: 100,

basePower: 0,

category: "Status",

desc: "Causes the target's Ability to become Insomnia. Fails if the target's Ability is Battle Bond, Comatose, Disguise, Insomnia, Multitype, Power Construct, RKS System, Schooling, Shields Down, Stance Change, Truant, or Zen Mode.",

shortDesc: "The target's Ability becomes Insomnia.",

name: "Worry Seed",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1, mystery: 1},

onTryHit(pokemon) {

const bannedAbilities = [

'battlebond', 'comatose', 'disguise', 'insomnia', 'multitype', 'powerconstruct', 'rkssystem', 'schooling', 'shieldsdown', 'stancechange', 'truant', 'zenmode',

];

if (bannedAbilities.includes(pokemon.ability)) {

return false;

}

},

onHit(pokemon) {

const oldAbility = pokemon.setAbility('insomnia');

if (oldAbility) {

this.add('-ability', pokemon, 'Insomnia', '[from] move: Worry Seed');

if (pokemon.status === 'slp') {

pokemon.cureStatus();

}

return;

}

return false;

},

secondary: null,

target: "normal",

type: "Grass",

zMove: {boost: {spe: 1}},

contestType: "Clever",

},

wrap: {

num: 35,

accuracy: 90,

basePower: 15,

category: "Physical",

desc: "Prevents the target from switching for four or five turns (seven turns if the user is holding Grip Claw). Causes damage to the target equal to 1/8 of its maximum HP (1/6 if the user is holding Binding Band), rounded down, at the end of each turn during effect. The target can still switch out if it is holding Shed Shell or uses Baton Pass, Parting Shot, Teleport, U-turn, or Volt Switch. The effect ends if either the user or the target leaves the field, or if the target uses Rapid Spin or Substitute successfully. This effect is not stackable or reset by using this or another binding move.",

shortDesc: "Traps and damages the target for 4-5 turns.",

name: "Wrap",

pp: 20,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

volatileStatus: 'partiallytrapped',

secondary: null,

target: "normal",

type: "Normal",

contestType: "Tough",

},

wringout: {

num: 378,

accuracy: 100,

basePower: 0,

basePowerCallback(pokemon, target) {

return Math.floor(Math.floor((120 \* (100 \* Math.floor(target.hp \* 4096 / target.maxhp)) + 2048 - 1) / 4096) / 100) || 1;

},

category: "Special",

desc: "Power is equal to 120 \* (target's current HP / target's maximum HP), rounded half down, but not less than 1.",

shortDesc: "More power the more HP the target has left.",

isNonstandard: "Past",

name: "Wring Out",

pp: 5,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Normal",

zMove: {basePower: 190},

maxMove: {basePower: 140},

contestType: "Tough",

},

xscissor: {

num: 404,

accuracy: 100,

basePower: 80,

category: "Physical",

shortDesc: "No additional effect.",

name: "X-Scissor",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: null,

target: "normal",

type: "Bug",

contestType: "Cool",

},

yawn: {

num: 281,

accuracy: true,

basePower: 0,

category: "Status",

desc: "Causes the target to fall asleep at the end of the next turn. Fails when used if the target cannot fall asleep or if it already has a major status condition. At the end of the next turn, if the target is still active, does not have a major status condition, and can fall asleep, it falls asleep. If the target becomes affected, this effect cannot be prevented by Safeguard or a substitute, or by falling asleep and waking up during the effect.",

shortDesc: "Puts the target to sleep after 1 turn.",

name: "Yawn",

pp: 10,

priority: 0,

flags: {protect: 1, reflectable: 1, mirror: 1},

volatileStatus: 'yawn',

onTryHit(target) {

if (target.status || !target.runStatusImmunity('slp')) {

return false;

}

},

condition: {

noCopy: true, // doesn't get copied by Baton Pass

duration: 2,

onStart(target, source) {

this.add('-start', target, 'move: Yawn', '[of] ' + source);

},

onResidualOrder: 19,

onEnd(target) {

this.add('-end', target, 'move: Yawn', '[silent]');

target.trySetStatus('slp', this.effectData.source);

},

},

secondary: null,

target: "normal",

type: "Normal",

zMove: {boost: {spe: 1}},

contestType: "Cute",

},

zapcannon: {

num: 192,

accuracy: 50,

basePower: 120,

category: "Special",

desc: "Has a 100% chance to paralyze the target.",

shortDesc: "100% chance to paralyze the target.",

name: "Zap Cannon",

pp: 5,

priority: 0,

flags: {bullet: 1, protect: 1, mirror: 1},

secondary: {

chance: 100,

status: 'par',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

zenheadbutt: {

num: 428,

accuracy: 90,

basePower: 80,

category: "Physical",

desc: "Has a 20% chance to flinch the target.",

shortDesc: "20% chance to flinch the target.",

name: "Zen Headbutt",

pp: 15,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 20,

volatileStatus: 'flinch',

},

target: "normal",

type: "Psychic",

contestType: "Clever",

},

zingzap: {

num: 716,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 30% chance to flinch the target.",

shortDesc: "30% chance to flinch the target.",

name: "Zing Zap",

pp: 10,

priority: 0,

flags: {contact: 1, protect: 1, mirror: 1},

secondary: {

chance: 30,

volatileStatus: 'flinch',

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

zippyzap: {

num: 729,

accuracy: 100,

basePower: 80,

category: "Physical",

desc: "Has a 100% chance to raise the user's evasion by 1 stage.",

shortDesc: "Goes first. Raises user's evasion by 1.",

isNonstandard: "LGPE",

name: "Zippy Zap",

pp: 10,

priority: 2,

flags: {contact: 1, protect: 1},

secondary: {

chance: 100,

self: {

boosts: {

evasion: 1,

},

},

},

target: "normal",

type: "Electric",

contestType: "Cool",

},

}