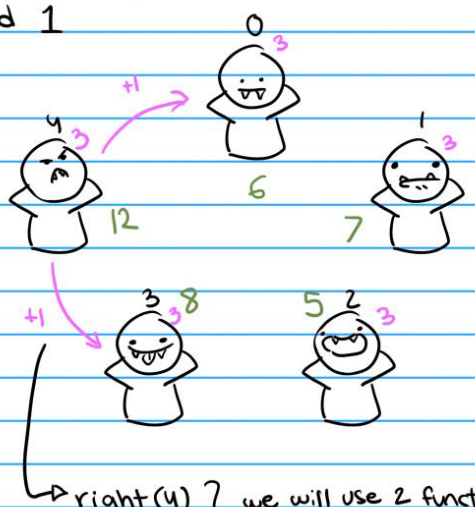
Game

● = the # they rolled ● = lifes

Round 1



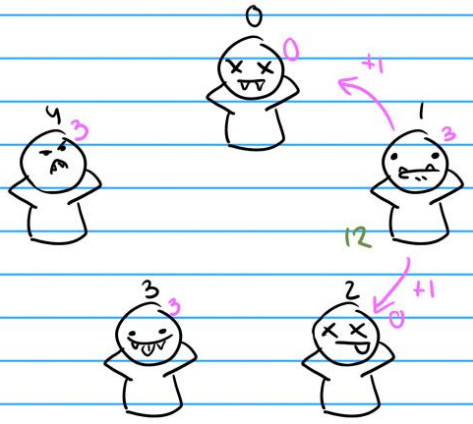
- 0 rolls 6
- lowest roll is 6 → id:0
- 1 rolls 7
- 2 rolls 5
- lr = 2,5
- 4 rolls 12
- 0 & 3 gain a life

right(4) } we will use 2 functions
left(4) } that will add lifes to the left & right of the vampire

rand() % 6 → gives a val 0-5

5+5 = 10 → midnight

Midnight Senario



- Vampire 1 roll a midnight
- V0 resercts
- V2 resercts

- V2 can roll now but V0s turn already passed so the don't get to roll

what is needed

- Array of names → ["Mary", "Sara", "Laila", ...]
- Array of lives → [3,3,3,...] ← all start with 3
- loop for each round
- Track lowest roll & who rolled it
  - ↓ save person who first rolled it
- Loop for entire game with exit condition
  - ↳ check to see if there is only 1 vampire left

This program will have some vampires play a garlic game where they roll some dice and every round 1 vampire loses a life if they roll the lowest number. The game ends when one vampire is standing

rules

- every round the vampire with the lowest roll gets -1 life
- if a vampire rolls a 12, the adjacent vampires gain +1 lifes
- game ends when 1 vampire is left

What is needed

- Array of names -> [“Mary”, “Sara”, “Laila”, ...]

- Array of lives -> [3, 3, 3, ...]
    - Everyone starts off with 3 lives

- Track lowest roll and who rolled it
    - save person who first rolled it
    - vars
        - LowestRollNameIndex
        - LowestRoll

- Loop for the entire game with an exit condition
    - exit condition
        - Check to see if there is only 1 vampire left

Pseudo code + structure

- ask for seed
- if the seed is not a number or out of range, end the program with an error message
- ask for number of vampires
- if the number of vampires is out of range, end the program with an error message
    - range is 2-10 inclusive
- Loop for each round
        - loop to iterate through vampires
            - if vampire has 0 lives
                - skip
            - vampire rolls
            - if roll is less than LowestRoll
                - name = [vampire name]
                - roll = [current roll]

            - if roll is a midnight (12)
                - call left([vampire index])
                - call right([vampire index])
                - (gives a life to the vampires adjacent to the current vampire)
        - take a life from the vampire with the lowest roll
        - if 1 vampire is left, break out of loop
        - set LowestRoll to 0
        - set LowestRollName to 0
            - even though this indexes the first vampire, this shouldn't be a problem because unless the first vampire rolls the lowest, the number will be set to something else during the round
- declare winner

UPDATED

Pseudo code + structure

- ask for number of vampires
- if the number of vampires is out of range, end the program with an error message
    - range is 2-10 inclusive
- ask for seed
- if the seed is not a number or out of range, end the program with an error message
- set the seed
- fill the array of lives with 3 lives each
- Loop for each round until 1 playing vampire is alive
    - set LowestRoll to a number higher than the possible highest roll number (so like 11)
    - loop to iterate through playing vampires (index that is less than the size)
        - if vampire has 0 lives
            - skip
        - vampire rolls
        - if roll is less than LowestRoll
            - lrvi = [vampire name]
            - lowestRoll = [current roll]
        - if roll is a midnight (12)
            - call left([vampire index])
            - call right([vampire index])
            - print that they sparkle if their lives are greater than 1
            - print that they are resurrected if their lives are less than 1 (so 0)
            - give a life to the vampires adjacent to the current vampire by incrementing their index in the lives[] array
    - take a life from the vampire with the lowest roll
    - if 1 vampire is left, break out of loop
- declare winner by finding whoever is left through the getFirstLivingVP() function

- function to roll
    - generate a random number from 0-5 and return it
- function to get number of alive vampires
    - crates a value int vampiresAlive
    - iterates through the playing vampires and increments vampiresAlive if the vampire has more than 0 lifes
    - return vampiresAlive

- function to get first living vampire
    - increment though playing vampires until a vampire with more than 0 lives is found
    - returns that vampire's index
- function to get player to the left
    - return (the vampire index + the total number of playing vampires -1)%the total number of playing vampires
    - we add the total number of playing vampires so that we don't index -1 by accident
- function to get player to the right
    - return (the vampire index + 1)%the total number of playing vampires
- function to print if a vampire sparkles or resurrects
    - if they have >0 lifes
        - print they are resurrected
    - else (they have more than 0 lifes)
        - print they sparkle