Project 9 - Cookie Clicker

-A game which utilizes a new variable type in Jack (Hershel Thomas and Livia Wyler)

Concept / Idea:

Our Cookie Clicker game is a classic incremental game where the player starts with a small amount of cookies, then presses the space key for more cookies, and then buys upgrades to increase their cookie production, aiming for the most cookies possible. The core gameplay loop involves clicking the cookie, purchasing upgrades that provide passive cookie generation, and gradually unlocking the entire game.

Architecture:

- **Main.jack:**
    - **Purpose:** This is the entry point of the application
    - **Logic:** It creates an instance of the CookieClicker class and starts the main game loop, then disposes of the CookieClicker object when the game loop is over. It is responsible for managing the start and end of the game.
- **CookieClicker.jack:**
    - **Purpose:** This class encapsulates the main game logic and state. It is the central manager of the game.
    - **Logic:**
        - **Initialization:** Sets up initial game variables like the number of cookies, cookies per click, cookies per second, upgrade objects, keyboard mappings, timers and position of the cookie.
        - **Game Loop:** The *run()* method contains the main game loop that handles:
            - **Input:** Reads keyboard input from the user.
            - **Clicking:** Updates the cookie count when spacebar is pressed, along with a visual change to indicate it was pressed.
            - **Upgrades:** Purchases upgrades when the corresponding keys are pressed and the user has enough cookies.
            - **Cookies per Second:** Updates the number of cookies based on passive upgrades over time.
            - **Drawing:** Handles the cookie visual feedback.
        - **De-allocation:** Disposes of the class once the loop exits.
- **LongNum.jack:**
    - **Purpose:** Implements custom *LongNum* class. To allow large numbers to be stored.
    - **Logic:** Contains logic for basic calculations: addition, subtraction, division, multiplication, getting the right/left sides, string conversion for printing to screen comparing numbers using greater than, less than, equal to, and the dispose method to de-allocate memory.
        - **Initialization:** The genius behind the new variable class. Stores a leftInt and rightInt in order to generate larger numbers. When the rightInt reaches 10,000, the leftInt is incremented by 1 and rightInt is reset to 0. This allows two ints to be stored in a combination to get numbers in the ten millions place.
        - **Math Expression:** Manages how/when to increment/decrement the *leftInt* and *rightInt.* This allows math and expressions to be used.
- **LongNumbers.jack**:
    - **Purpose:** A class to store common long numbers in the game.
    - **Logic:** Initializes all basic numbers such as 0, 1, 2, 7, 15, etc. All functions of the class return a *LongNum* object that can be used in other classes, and also includes a dispose method to de-allocate the class.
- **Upgrades.jack**:.
    - **Purpose:** Represents a single upgrade type. Manages the cost, amount owned, and Cookies-per-Second (CPS) generation.
    - **Logic**:
        - **Initialization:** Sets the initial cost, amount, and CPS for the upgrade.
        - **getUpgrade():** Increments the amount of upgrade purchased, and increases the cost of the upgrade by a defined amount.
        - **Getter Methods:** Includes methods to return the cost, amount, and CPS of the upgrade.

- **CookieDrawings.jack:**
  - **Purpose:** Handles the drawing of the cookie onto the screen.
  - **Logic:** Contains two functions, one to draw a big cookie (filled in) and a big pressed cookie (not filled in). It does this by using memory pokes to update the memory addresses of the screen.
- **UpgradeDrawings.jack:**
  - **Purpose:** Handles the drawing of the different upgrade icons.
  - **Logic:** Contains multiple methods for each different upgrade. Each method has a series of memory pokes to update the memory address for the screen to draw the associated icon.
- **Scoreboard.jack:**
  - **Purpose:** Handles the display of text information for the user.
  - **Logic:** Contains various functions to print the initial text of the game, printing the number of cookies, printing cookies per second, printing upgrades and all related text for the game. Also calls upon *UpgradeDrawings* to draw icons for upgrades.

Motivation:

The motivation for the game came upon us when we wanted to figure out a way to store and manipulate very large numbers, far beyond the 34,000 integer limit in Jack. We realized that a game that would use this new implementation would require a generation of very large numbers quickly. Thus, to showcase this implementation, we created an addictive Cookie Clicker game which overtime generates very large numbers.

Google Drive:

https://drive.google.com/file/d/1hvSN-PyIOR6KuWXssNFAZIvjTeOv1vZ0/view?usp=sharing

Names and Emails:

Hershel Thomas - hershel.thomas@post.runi.ac.il
Livia Wyler - livia.wyler@post.runi.ac.il